Recognition and Manipulation of Deformable Objects Using Predictive Thin Shell Modeling

Yinxiao Li

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2016

ABSTRACT

Recognition and Manipulation of Deformable Objects Using Predictive Thin Shell Modeling

Yinxiao Li

This thesis focuses on the task of dexterous manipulation of deformable objects, and in particular, clothing and garments. The task of manipulating deformable objects such as clothing can be broken down into a series of sub-tasks: (1) perceive and pick up garment, and then identify garment and recognize its pose; (2) using a manipulation strategy, regrasp the object to put it into a canonical state; (3) scan the surface of the object to find wrinkles, and use an iron to remove the wrinkles; (4) starting from the wrinkle-free state, fold the garment according to pre-planned sequence of manipulations with optimized trajectories; In this thesis, we will address all the phases of this process.

A key contribution of the work is innovative use of simulation. We use offline simulation results to predict states of deformable objects (i.e. cloth, fabric, clothing) that are then recognized by a robotic vision/grasping system to correctly pick up and manipulate these objects. The recognition will use the simulation engine to deform the models in real time to find correct matches. The simulation will also be used to find the optimized trajectories for the manipulation of the garments, such as the garment folding.

Table of Contents

List of Figures

List of Tables

Acknowledgments

The three-year Ph.D. life is an invaluable time period in my life, and will greatly guided me into a more mature person both in my life and study. This will not happen without the guidance, encouragement, and supports from my mentors, friends, and family.

My deep gratitude goes first to my advisor, Professor Peter Allen, who expertly guided me through my Ph.D. research, and shared excitement of years of discovery. His unwavering enthusiasm in pushing robotics research to a new level kept me constantly fully engaged with my projects all the time, and his personal generosity made my time at Columbia more enjoyable. Professor Allen taught me several essential skills, which led me to a productive researcher, especially for the intuition behind a difficult task. The intuition helped me to accurately break down an complicated question, and approach each component in a very efficient way. In addition, his open-minded attitude allowed me to have multiple interns to gain more hands experiences from the industry, which helped me to build up a solid career path.

Meanwhile, I would like to thank my dissertation committee members, Professor Eitan Grinspun, Professor Shih-Fu Chang, Professor Matei Ciocarlie, and Dr. Stan Birchfield, for attending my Ph.D defense and providing valuable feedbacks. Also, I would like to thank other professors and doctors who have offered valuable suggestions to my research during my Ph.D study: Professor Peter Belhumeur, Professor Shree Nayar, Dr. Yonghao Yue, Professor Changxi Zheng, and Professor Daniel Hsu.

My twice internships at Google in 2014 and 2015 summer also offers me tremendous industrial experience. The latest one was with my mentor Mei Han at Google Research. I can still remember the moment when I spent almost two days to find a bug in the code, and shared my excitement with my mentor close to the mid-night! The previous one was with Li Zhang and Steve Seitz at Google Seattle, where we developed a prototype app called “SmartShutter”. Taking lots of facial pictures to test the algorithm also brought lots of fun to us! I also received generous help from my intern

collaborators: Zhen Li, Jingyu Cui, and Yangqing Jia.

It has been lots of pleasure to work with my friends in Columbia Robotics Lab. I would like to thank Dr. Jon Weisz for helping me with setting up the Baxter robot, such as the calibration. Also, I really enjoyed the discussion with Jake Varley on deep learning related questions, as well as many interesting topics along the IROS 2015 trip. I would like to thank Dr. Hao Dang and Dr. Austin Reiter for the helps when I just started my Ph.D. program. Even though we only have one semester overlap, their suggestion made my life easier. In addition, I would like to thanks my friends at Columbia, who brought encouragement and joy to my life at here: Jiongxin Liu, Yan Wang, Jie Feng, Dingzeyu Li, Yun Fei, Timothy Sun, Jing He, Bingyi Cao, Thomas Berg, Adrian Tang, Daniel Miau, Chun-Yu Tsai, and Fang-hsiang Su.

I also would like to thank those students who have worked with me during the past three years: Danfei Xu, Xiuhan Hu, Michael Case, and Chih-Fan Chen. Without their contributions on the project, the progress of it could not be so fast. I still remember the moment when we worked together on the experiments, and recorded multiple video clips before the paper deadlines.

Having staying at the computer science department for three years, I have received warm supports from the staff members from the department. Here, I would like to extend my special thanks to Anne Fleming, Jessica Rosa, Cindy Walter, Daisy Nguyen, Remi Moss, and Elias Tesfaye.

To my parents.

Chapter 1

Introduction

1.1 Problem Statement

Imbuing robots with the ability to handle deformable objects is a challenging goal, and research in this area is increasing rapidly. Grasping and manipulation of deformable objects presents a host of new research challenges that are much more demanding than for rigid objects. A particular challenge is to fully understand the physics of deformation and to model deformable objects in a way that can be used by real robotic systems to manipulate the deformable objects in the presence of noise and uncertainty and with real-time constraints. This is a new area which hasn't been well explored by the researchers. Understanding of the physics of the deformation of the objects will greatly benefit both in recognition and manipulation of an AI agent, such as a robot. Physical-based modeling will create a bridge between the simulation world and the real world. We further specify our targeted objects as a set of deformable garments like a long-sleeve shirt or a pair of pants. The entire pipeline of manipulation of deformable objects, as shown in Figure 1.1, can be summarized as recognition, regrasping and unfolding, ironing, and folding. Each step will benefit from the physical-based modeling with a higher accuracy. The applied methods can then be extended to other similar deformable objects or tasks.

1.1.1 Recognition

Deformable objects such as clothes, fabrics, paper, and things that are soft, are ubiquitous in our daily life. In order to enable robots to manipulate deformable objects efficiently and effectively,

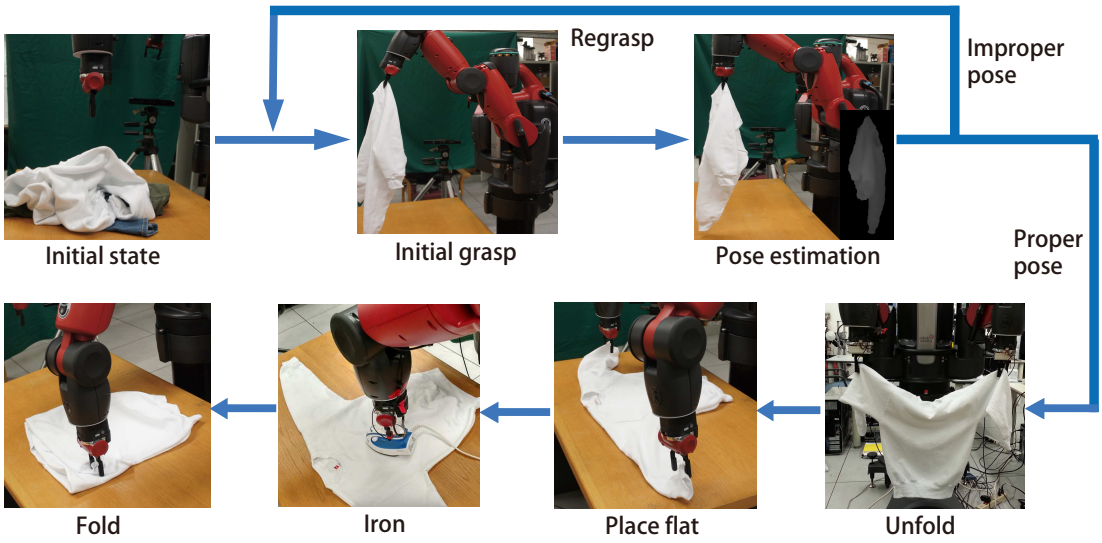


Figure 1.1: The entire pipeline of robotic manipulation of deformable objects.

we first have to let the robots recognize them through perception. Compared with rigid object recognition which has finite state spaces, deformable object recognition is much harder because of the very large dimensional spaces. Each space may have completely different appearance in terms of the material and pose of the object. More specifically, deformable objects are much harder to recognize and manipulate because of the large variance of appearance in terms of materials and the way they deform. This variance subsequently makes it difficult to establish a robust recognition pipeline to predict the *pose* of the deformable objects based on traditional visual sensors, such as optical cameras. For example, in robot grasping, it is possible to identify the grasping point location on a rigid object by a shape matching algorithm from a known object in a training set. However, due to the large number of states of a deformable object, it is much more difficult to identify the pose and grasping point.

1.1.2 Unfolding

With predicted pose and grasping point, we can proceed to bring the deformable object from a random pose to a desired pose by sequential manipulations such as unfolding a garment. This is a much more difficult task because the states of a deformable object are not easy to track and predict owing to large-dimensional state spaces. For example, where to grasp in the next step needs to be figured out. Some previous work focused on regrasping at the lowest point at each time, which may

succeed in a small number of garments. The goal is that after several steps of regrasping, robot holds the garment at two desired positions. Taking a sweater as an example, we defined the optimal grasping positions on the two sleeves. Grasping at on a deformable object is also a challenge. Some materials such as a piece of denim, it usually has a big flat region when hanging under gravity. Therefore, during the regrasping, in terms of the opening of the gripper, the robot needs to find a better location for grasping, such as a surface bump on the sweater.

1.1.3 Wrinkle Detection and Ironing

A garment is typically manufactured by stitching together pieces of fabric, each of which is cut out from an originally flat fabric sheet. As long as the contacts between the weft and warp threads do not slide over each other, a regular garment is thus locally *developable*, meaning that the Gaussian curvature (the product of the two principal curvatures) is zero everywhere except along the seams. A common case is when a piece of cloth has been stretched and cannot be flattened on the table. When experiencing forces due to squeezing or swirling motions during laundry, the garment may undergo a series of deformations, resulting in (possibly a mixture of) the following two types of developable regions: (1) non-smooth regions; (2) smooth regions. Non-smooth regions usually cannot be flattened by iterative pulling. We define those regions as *permanent wrinkles*, which are the ones we want to use the iron on. For the smooth regions, most of them can be flattened by pulling the boundaries iteratively. We try to remove the permanent wrinkles through a series of techniques, using an iron mounted on one of the Baxter robot hands.

1.1.4 Folding

Robotic folding garments autonomously remains a challenging task. Even with a clear folding plan, the garment may not be folded as expected. Certain trajectories will cause the garment to move on the table, or create wrinkles on the garment. Other trajectories may fail altogether. Figure 1.2 shows a few failure examples with improper trajectories. We use green tape on the table to show the original position of the garments. The first two rows show that if the moving trajectory is too low and close to the garment, the folded part will fall down, pull the rest, and cause drift of the whole garment. These cases usually happen when the folding step is lengthy without trajectory optimization. The third row shows a case where the folding trajectory is too high, which will cause

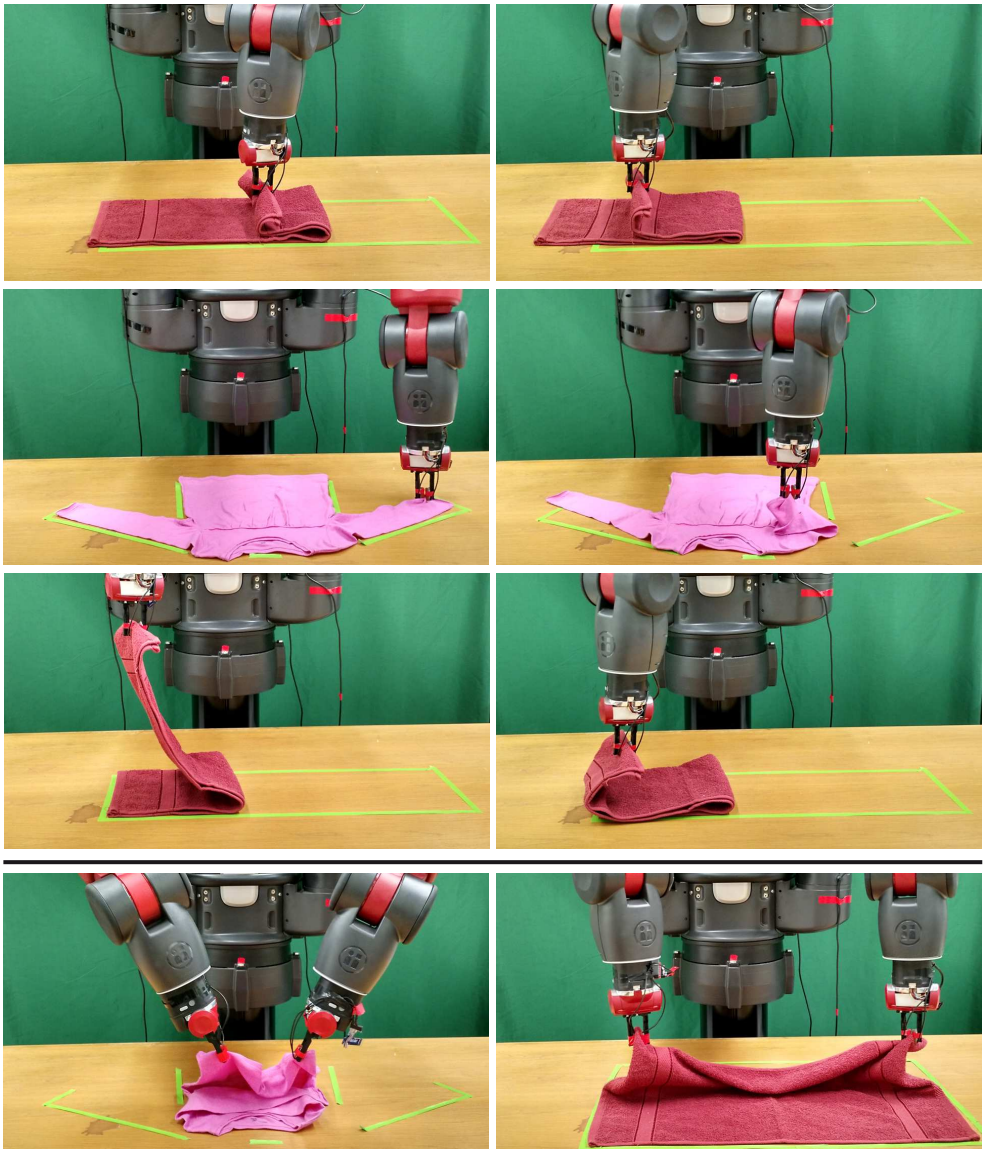


Figure 1.2: Failure example with improper folding trajectories. FIRST ROW: Folding trajectory is low and flat that causes drift to the towel and long-sleeve T-Shirt. SECOND ROW: Folding trajectory is too high when the gripper approaching the target folding position that piles up the towel. THIRD ROW: Dual-arm folding. If the distance between the two arms is too close, the folding may fail.

extra wrinkles or even piling up. The last row shows two cases using two arms to fold. If the arms are close, the part in between loses tension, and will fall down and pull the rest away. One of the solution is to create optimized trajectories for the folding that will overcome these problems.

1.2 Our Approach

Our initial interest is detecting the pose of deformable objects such as garments as a part of a larger pipeline for manipulating these objects. Once the robot has detected the pose of the objects, it can then proceed to manipulate those objects, for tasks such as garment folding. The deformable objects have a large dimensional space in terms of a huge number of ways they can deform. Therefore, it is difficult to track and recognize their categories and poses. We use offline simulation to predict states of deformable objects modeled as thin-shells (i.e. cloth, fabric, clothing) that can then be recognized by a robotic vision/grasping system to pick up and manipulate these objects. We exploit our knowledge and ability to accurately simulate the physics of deformation to simulate and predict how deformable objects behave.

To recognize the categories and poses of these deformable objects, we exploit our expertise in using data-driven methods to pre-compute and index deformable object models. We apply 3D shape features with a non-convex optimizer to learn the best weights for each class. Newly emerged low-cost depth sensors such as Microsoft Kinect can provide accurate and fast depth measurements. With this depth information, a robotic system is able to resolve the ambiguity of visual appearance better, and therefore provide higher performance on recognition tasks.

Further, we use a database, which contains a number of physics-based simulated deformable objects, and to store landmark features of objects that can be used for alignment and regrasping of the deformable objects. We will introduce a registration algorithm that combines rigid and non-rigid registration algorithm to find the correspondence between the database model and the reconstructed model. During the regrasping, to find a better location to grasp, the found location needs to be locally refined. One possible solution is to use a small range sensor to scan the surface of a garment, and compute the curvature to find the bump. With the desired grasping points, the robot can go ahead and place the garment flat on a table.

With the garment flattened on the table, we can detect the permanent wrinkles, which cannot be removed by pulling the boundaries, and try to remove them by robotic ironing. For a permanent wrinkle smaller than the iron base with its surrounding garment region being more or less flat, we found that we can reduce the permanent wrinkle by static placement of the iron on the surface or dynamic sliding of the iron onto the surface for larger ones. For a region with mixed deformations (i.e., smooth and non-smooth regions), we expect that we can pull the fabric to locally remove the

developable deformation (by using e.g., [Sun *et al.*, 2015]).

Finally, we also use the physics-based simulation to understanding fold, and how folding trajectories can be simulated, tested and used online. We emphasize that the testing will be with a real robotic system on real deformable objects (i.e. clothing garments), but that the simulation engine, used both offline and online, and the database will be the drivers of the system. Given the garment on the table, the contour and key points can be automatically detected. Then, a pre-defined folding plan can be generated to fold the garment. Autonomous garment folding by a robot has been investigated by researchers years ago with many different folding strategies. An interesting observation is that the layout of the same folding action can vary in terms of the material properties such as cloth hardness and the environment such as friction between the garment and the table. Given the starting and ending folding positions, different folding trajectories will lead to different results. We begin by assuming the garment is placed flat on the table initially, as shown in our previous work [Li *et al.*, 2015a]. By detecting the key points of the garment (see section 7.3.1), a pre-defined folding plan is used to create optimal trajectories for folding the garment. After several steps, we obtain a desired folding result in the real world using the Baxter robot, which is comparable to the result from the simulation. In between, trajectory optimization is important for garment folding. Bad trajectories will bring uncertainties to the layout given the same folding plan. One possible case is that the garment shifts on the table during one folding action so that the targeted folding position is also moved. Another case is that an improper folding trajectory causes additional deformation of the garment itself, which can accumulate.

In the following chapters, we will follow the entire pipeline of robotic manipulation of deformable objects (see Figure 1.1), and address all the mentioned methods above with experimental results.

1.3 Impact

This work creates an open source, extensible, 3D database for dissemination to the robotics and graphics communities. Data-driven methods are proliferating, and they need to be applied, tested and validated in real environments. This database has been served as infrastructure for developing advanced robotic machine learning algorithms. Within the database, each garment is fully simulated with a number of depth images and 3D mesh models for all the semantic labeled grasping points. We demonstrate that using the database, we can improve the recognition and the manipulation of deformable objects. One example is training from the simulated mesh models to recognition an unknown object by 3D shape-based features. The second one is applying the simulated mesh model to guide the iterative regraspings by the rigid and non-rigid registrations. The third one is importing the mesh model into the simulator and computing the optimized trajectories for manipulation of the deformable objects. Meanwhile, we extensively tested these methods with designed experiments such as garment recognition via picking up, unfolding it to a desire state, and folding it like human beings. Experimental results show that with the simulation database, the performance of the recognition and manipulation of deformable objects can be greatly improved.

We expect researchers in or close to the robotics area to use the database, and in particular, address the problem of learning on large-scale databases. To this end, we release the script for the entire simulation in Maya that we used to create the database. The simulation can be easily adapted to the new garment mesh models by giving the semantic grasping points. We also release the training and testing code for the volumetric approach for the recognition [Li *et al.*, 2014b], trajectory optimization in the Maya simulator for computing optimal trajectories, and key points localization for transferring manipulation between different domains [Li *et al.*, 2015b]. We sincerely hope the open source software can greatly benefit the research in this community and push it into a new height.

Chapter 2

Related Work

In the robotics and computer vision communities, recognition and manipulation of deformable objects such as garments, are well-known challenging tasks. In recent years, mature solutions to manipulation of rigid objects have emerged and been applied in industry [Brogrdh, 2007]. However, in the fabric and food industry, which involve a large number of deformable objects, there is still a large gap between the high demand for automatic operations, and the lack of reliable solutions.

There are several research groups working on the recognition and manipulation of deformable objects. Here, we summarize their work in terms of several subjects such as end-to-end system, types of vision sensor, etc., as shown in Table 2.1. Most of the groups use a database of garments for the recognition task. All of them use depth information for recognition such as a Kinect sensor or a pair of stereo cameras. Some of them use built robots such as PR2 or Baxter, while others use industrial robotic arms. The methods for recognition are different which bring variance in recognition speed. Within the listed methods, [Li *et al.*, 2014b] reports the fastest speed (less than 1 sec) in the recognition task. We also check if each research group has a complete garment manipulation pipeline, which includes recognition, unfolding, ironing, and folding.

2.1 Recognition of Deformable Objects via Depth Sensor

2.1.1 Recognition via Depth Perception

A significant amount of previous research has focused on deformable object detection. In these works, the primary goal is to detect the deformable objects in the scene via some features, but they

	UC Berkeley	Clopema	Columbia	Osawa, et. al	Bosch LLC	Clemson	Kita, et. al
Database for recognition	√ (7)	√ (24)	√ (37)		√ (1)	√ (30)	
Garment simulation	√		√				√
Kinect sensor	√	√	√			√	
Stereo camera		√		√	√		√
Robot	PR2 robot	Dual YASKAWA arm	Baxter robot	Dual Mitsubishi Denki RV- M1	PR2 robot	1 Puma arm	Single arm
Garment state recognition method	Hidden Markov model	Active Random Forest	Sparse coding & 3D shape matching	Image matching	Pre-labeled fiducials	Bag-of-words	3D shape matching
Recognition speed	>30s	8~10s	<1s	7s	>60s	2~3s	5-20s X 8
Garment Unfolding	√	√	√	√	√	√	
Robotic Ironing			√				
Garment Folding	√	√	√	√	√		√

Table 2.1: Comparison of systems of robotic recognition and manipulation of deformable objects. For the row of database for recognition, the number indicates the number of different garments in the database.

do not explicitly compute object pose. Most of the techniques use part-based or contour-based models. Ravishankar, et al. [Ravishankar *et al.*, 2008] proposed a multi-stage contour based template matching method of detecting deformable objects. However, this method may fail if there is no explicit contour detected. Felzenszwalb, et al. [Felzenszwalb *et al.*, 2010] described an algorithm based on mixtures of multi-scale deformable part models trained using a discriminative procedure, which achieved a higher detection accuracy on PASCAL challenges. Another fast deformable object detection approach was proposed by Pedersoli, et al. [Pedersoli *et al.*, 2011], which improved the part-based model by doing a multiple-resolution hierarchical layer structure which increased the detection accuracy. Though their work has very promising detection accuracy on many different object categories, it still does not address the pose recognition problem.

There has been previous work on the recognition and manipulation of deformable objects. B. Willimon, et al. [Willimon *et al.*, 2013] [Willimon *et al.*, 2011] used interactive perception to classify the clothing type. However, their work basically focused on small clothing such as socks and short pants which usually consist of a single color. The proposed method may fail with some

fully textured clothes because it relies on a color-based image segmentation algorithm. Miller, et al. [Miller *et al.*, 2012], Wang, et al. [Wang *et al.*, 2011], Schulman, et al. [Schulman *et al.*, 2013], Cusumano-Towner, et al. [Cusumano-Towner *et al.*, 2011] have done some impressive work in clothing recognition and manipulation. They have successfully enabled the PR2 robot to fold clothing and towels. Their methods mainly focus on aligning the current edge/shape from observation to an existing shape, which may fail when the clothes shape is uninformative. A series of works on clothes pose recognition were done by Y. Kita, et al. [Kita *et al.*, 2011a] [Kita *et al.*, 2011b] [Kita and Kita, 2002]. Their work demonstrated the ability to identify the pose of the clothes by registration to pre-recorded template images. Without showing more statistical results, it is not clear that the proposed algorithm works on a random grasping point over the entire target clothing.

The pose estimation problem can be formulated as an image classification task aiming at depth images of the target garment. The idea of bag-of-feature (BOF) can be employed here to solve this problem. The BOF model has turned out to be one of the state-of-the-art approaches in dealing with image classification tasks. There are several noticeable works done in the past years. Fei-Fei and Perona [Fei-Fei and Perona, 2005] proposed a Bayesian hierarchical model for image classification using BOF model, which was widely used in natural scene classification. Grauman and Darrell [Grauman and Darrell, 2005] introduced an efficient pyramid matching kernel which dramatically sped up the feature matching process. Because the traditional BOF model discards the spatial layout of the features in an image, a spatial pyramid matching (SPM) approach was proposed by Lazebnik, et al [Lazebnik *et al.*, 2006] to overcome the spatial loss, which greatly improved the classification accuracy. In addition to that, Yang et al. [Yang *et al.*, 2009] improved the SPM by generalizing vector quantization to sparse coding followed by spatial max-pooling and a linear-kernel SVM. The proposed method outperformed a few previous methods with a relatively high speed. In our approach, we employed Yang's idea with modifications to some specific cases.

2.1.2 Recognition via Volumetric Approach

While most of the recognition methods are based on optical sensors including single or stereo cameras. In contrast, our most recent work [Li *et al.*, 2014a] on recognition and pose estimation of deformable clothes uses a Kinect depth sensor. Our method does recognition on individual depth images and uses majority voting to get a comprehensive result. When the deformation is com-

plicated, the inherent noise in the Kinect's sensor and the missing data due to the obstruction of self-occlusions may confuse the algorithm. The method relies on processing a large number of depth images to find matching grasping points. However, running the whole pipeline on hundreds of input images slows down the method and thus limits the applicability. If the experimental setting is without an advanced computing resource such as a powerful graphics card, this method can be a desired one.

With powerful computing resources, reconstructing a 3D model of the garment, and use that to search a pre-computed database of simulated garment models in different poses can be more accurate and efficient. With the increasing popularity of Kinect sensor, there are various methods emerging in computer graphics such as KinectFusion and its variants [Newcombe *et al.*, 2011][Chen *et al.*, 2013][Li *et al.*, 2013]. Although these methods have shown success in reconstructing static scenes, they do not fit our scenario where a robotic arm is rotating the target garment about a grasping point. Therefore we first do a 3D segmentation to get the masks of the garment on the depth images, and then invoke KinectFusion to do the reconstruction.

Shape matching is another related and long-standing topic in robotics and computer vision. On the 2D side, various local features have been developed for image matching and recognition [Huttenlocher *et al.*, 1993][Latecki *et al.*, 2000][Lowe, 1999], which have shown good performance on textured images. Another direction is shape-context based recognition [Belongie *et al.*, 2002][Toshev *et al.*, 2010][Tu and Yuille, 2004], which is better for handwriting and character matching. On the 3D side, Wu *et al.* [Wu *et al.*, 2008] and Wang *et al.* [Wang *et al.*, 2006] have proposed methods to match patches based on 3D local features. They extract Viewpoint-Invariant Patches or the distribution of geometry primitives as features, based on which matching is performed. Osada and Funkhouser [Osada and Funkhouser, 2001], Thayananthan *et al.* [Thayananthan *et al.*, 2003], and Frome *et al.* [Frome *et al.*, 2004] apply 3D shape-context as a metric to compute similarities of 3D layout for recognition. However, most of the methods are designed for noise-free human-designed models, without the capability to match between the relatively noisy and incomplete mesh model produced by Kinect and the human-designed models. Our method is inspired from 3D shape context [Frome *et al.*, 2004], but provides the capability of cross-domain matching with a learned distance metric, and also utilizes a volumetric data representation to efficiently extract the features.

2.2 Unfolding of Deformable Objects

Osawa *et al.* [Osawa *et al.*, 2007] proposed a method using dual-arm to unfold a garment from pick up. However, from the examples in the paper, this method requires a garment with unique color and a clean background for color-based image segmentation. The PR2 robot is probably the first robot that has successfully unfolded deformable objects such as a towel or a T-shirt [Maitin-Shepard *et al.*, 2010]. The visual recognition in this work targets corner-based features, which may not exist in many soft garments. The subsequent work has improved the prediction of the state of a garment using a HMM framework by regrasping at the lowest corner point [Cusumano-Towner *et al.*, 2011]. However, this method also requires a clean background, and thus limits the applicability.

The work that is closest to ours is by Doumanoglou *et al.* [Doumanoglou *et al.*, 2014]. This work has impressive results for unfolding of a number of different garments. They use a dual-arm industrial robot to unfold a garment guided by a set of depth images which provide a regrasping point. This method achieves promising accuracy. A major difference between this work and ours is our use of simulated predictive thin shell models for garments to automatically create a large database of garments and their poses. Their training set is a number of physical garments that have been grasped at different grasping points to create feature vectors for learning. Given the physical nature of this training set, it can be very time-consuming to create, and may have problems encompassing a wide range of garments and different fabrics which we can more easily accommodate in the simulation environment. We also use an online registration of a reconstructed volumetric mesh with the simulated model to find regrasping points. By this method, we can choose arbitrary regrasping points without having to train the physical model for the occurrence of the grasping points. This allows us to choose any point on the garment at any time as the regrasping point. We have also developed a closed loop tactile feedback algorithm that allow us to perform a local search at the regrasping point that alleviates the instability of the grasp during regrasping as mentioned in [Doumanoglou *et al.*, 2014].

In our method, the regrasping point is located by mapping the pre-determined point from simulation mesh to the reconstructed mesh. Therefore, a fast and accurate registration algorithm plays a key role in our method. Rigid or non-rigid surface registration is a fundamental tool to find shape correspondence. A thorough review can be found in [Tam *et al.*, 2013]. Our registration algorithm builds on previous techniques for rigid and non-rigid registrations. First, we use an iterative closest

point method [Besl and McKay, 1992] to rigidly align the garment. We use a distance field to accelerate the computation. Next, we perform a non-rigid registration to improve the matching by locally deforming the garment. Similar to [Li *et al.*, 2008], we find the correspondence by minimizing an energy function that describes the deformation and the fitting.

2.3 Ironing Deformable Objects

When we have a garment placed flat on a table, there are may exist some wrinkles on the surface, which we do not want to see after the folding. In our daily life, people use iron to remove the wrinkles on the garment, such as ironing a shirt. In recent years, a few researchers start to solve the ironing problem by using a robot. To have an efficient ironing process, and identifying wrinkles is the first step of it. To compute the surface curvature, many researchers proposed methods to extract features from depth visual cues [Maitin-Shepard *et al.*, 2010][Ramisa *et al.*, 2012][Willimon *et al.*, 2013]. Those detected wrinkles are useful for robotic grasping, regrasping, and object classification. One notable work that uses detected wrinkles to achieve a flattened surface is by Sun *et al.* [Sun *et al.*, 2015]. They apply two SLR cameras as a stereo pair to reconstruct a high quality depth map of garment surface. Then by estimating the volume of the ridges, the robot arms are able to flatten the garment by iterative dragging.

Robotic ironing of deformable garments is a difficult task primarily because of the complex surface analysis, regrasping, and hybrid force/position control of the iron. Without wrinkle detection, Dai *et al.* introduced an ironing plan that spreads out the whole garment surface by dividing it into several functional regions [Dai *et al.*, 2004b]. For each region, in terms of the size and shape, an ironing plan is automatically generated. Dai *et al.* also addressed the ironing problem considering the folding lines [Dai *et al.*, 2004a].

Learning from demonstration is an effective approach to transfer knowledge and teach a robot to do a specific task [Argall *et al.*, 2009]. Kormushev *et al.* enabled the robot to learn positional and force skill from human demonstration, and then encoded them into the controller for further controlling of the robotic arms [Kormushev *et al.*, 2011]. Calinon *et al.* also addressed similar problem by using several examples of the skill demonstration through kinesthetic teaching, incorporated with safety constraints [Calinon *et al.*, 2010].

2.4 Folding Deformable Objects

With the garment fully spread on the table, attention is turned to parsing its shape. S. Miller *et al.* have designed a parametrized shape model for unknown garments [Miller *et al.*, 2011][Miller *et al.*, 2012]. Each set of parameters defines a certain type of garment such as a sweater or a towel. The goal is to minimize the distance between the observed garment contour points and points from the parametrized shape model. The fitting score between the observed contour and the shape models can also be used for recognition of garment category. However, the average time for the fitting procedure is $30-150$ seconds and sometimes does not converge. The contour-based garment shape model was further improved by J. Stria *et al.* using polygonal models [Stria *et al.*, 2014a]. The detected garment contour is matched to a polygonal model by removing non-convex points using a dynamic programming approach. The landmarks on the polygonal model is then mapped to the real garment contour, and followed by the generation of the folding plan.

Folding is the ultimate goals of garment manipulation and only a few researchers have achieved this. F. Osawa *et al.* used a robot to fold a garment with a special purpose table that contains a plate that can bend and fold the clothes assisted by a dual-arm robot. The robot mainly worked on repositioning the clothes for the plate for each folding action. Within several "flip-fold" operations, the garment can be folded. Another folding method using a PR2 robot was implemented by J. van den Berg *et al.* [van den Berg *et al.*, 2010]. The core of their approach was the geometry reasoning with respect to the cloth model without any physical simulation. Contour fitting at each step took relatively longer than execution of the folding actions, which reduced its efficiency. This was further sped up by J. Stria *et al.* [Stria *et al.*, 2014b] using two industrial arms and a polygonal contour model. They showed impressive folding results by utilizing a specifically-designed gripper [Le *et al.*, 2013] that is suitable for cloth grasping and manipulation.

Part I

Recognition of Deformable Objects

Chapter 3

Recognition via Depth Perception

Figure 1.1 shows an overview of our pipeline for dexterous manipulation of deformable objects. The first step is the visual recognition of deformable objects. We need to have a large set of exemplars of how garments will look visually when arbitrarily grasped. Physically having people, or a robot arm, successively pick up an object and image its appearance is too time consuming and cannot span the large space we are hoping to learn. However, using advanced simulators such as *Maya* to physically simulate deformable objects, we can produce thousands of exemplars efficiently, which we can then use as a corpus for learning about how the deformed garments will appear.

There are two phases in the recognition. In the first phase, we train on a previously computed database of deformed objects. From that we build a recognition codebook that can be used in the second phase to first recognize the category and then the pose of an object through a two-layer classifier that we have developed. The framework is discussed in detail below.

3.1 Simulating Deformable Objects

We have developed an off-line simulation process whose results can be used to predict poses of deformable objects. The off-line simulation is time efficient and more accurate compared with acquiring data via sensors from real objects. In the off-line simulation, we use a few well-defined garment mesh models such as sweaters, jeans, and short pants, etc. Similar garment mesh models can also be obtained from Poserworld and Turbo Squid. We also can generate models by using "*Sensitive Couture* " software [Umetani *et al.*, 2011]. Figure 3.4 shows a few of our current garment

models rendered in *Maya* software.

In *Maya*, a mesh model can be converted into an *nCloth* model which can be then simulated with some cloth properties such as hanging and falling down. Some advanced features in *Maya* may also be applied, including cloth thickness and deformation resistance, etc.. In addition, any vertex on the mesh can be selected as a constraint point to simulate a draping effect. The hanging under gravity effect of the garment models is shown in Figure 3.4, (a) and (b) are shown Figure 3.4, (c) and (d). Figure 3.1 shows a small sample of different picking points of a single garment hanging under gravity that simulated in *Maya*.

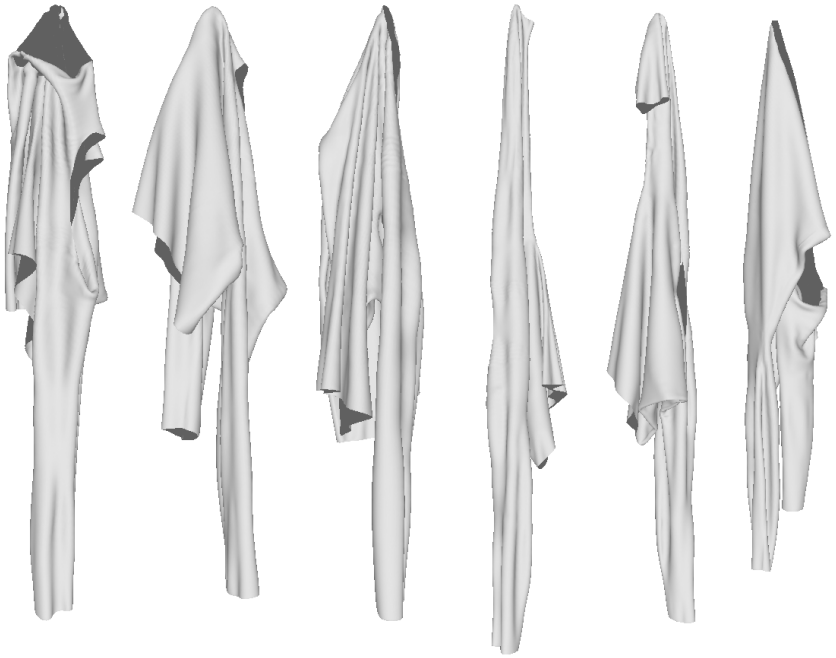


Figure 3.1: 6 different picking points of a sweater garment mesh model simulated in *Maya*.

3.2 Generating Training Exemplars

In *Maya*, virtual cameras can be set up to capture depth images of the rendered object. In the real world, a garment can be picked up by a robot with various view points from the camera. So it is meaningful to build up a geodesic dome(see Figure 3.2) with cameras placed on it that capture the images from various view points. Our previous work [Goldfeder *et al.*, 2009] has shown that depth images can help to recognize object categories via feature descriptors and clustering algorithms.

Also, we use 3D depth images instead of 2D color images which rely on texture rather than the shape. In our off-line simulation, we set up a group of 90 cameras on a geodesic dome with different view points and positions. Figure 3.2 shows a lay out of the 90-camera system with a sweater mesh model placed in the center.

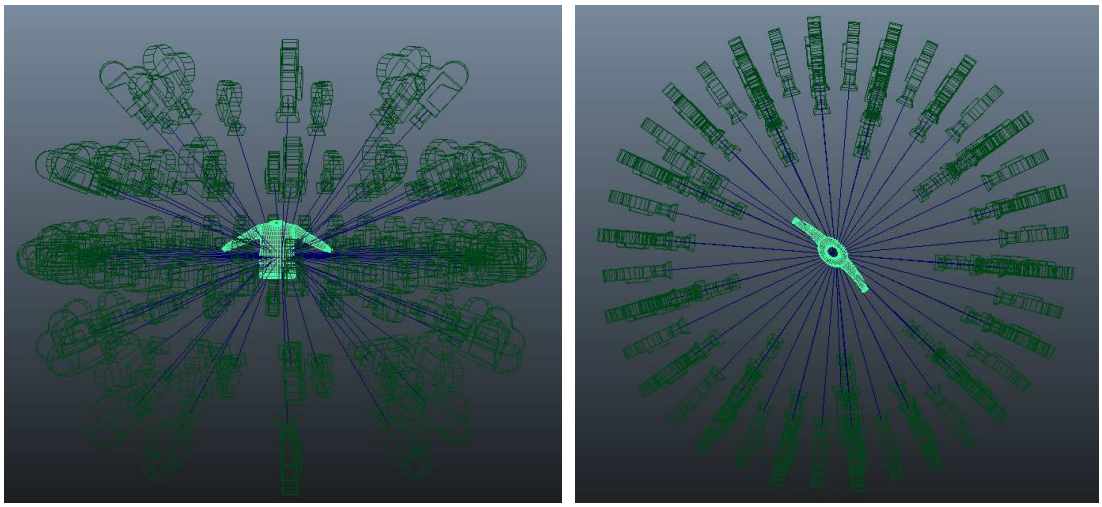


Figure 3.2: LEFT: 90-virtual-camera system viewed horizontally. RIGHT: 90-cameras system viewed from the top.

In our experiments, the depth images were captured via the 90 cameras. The workflow of getting the depth images can be summarized as the followings.

1. For each input garment model, a set of $20 - 50$ grasping points is predefined in terms of the garment categories, e.g., sweater or pant.

2. For each of the grasping points, a simulation of draping under gravity is carried out and rendered when the garment achieves a stable state (e.g. no shaking).

3. The 90-cameras system will capture depth images of the model from different view points.

Please note that all the 90 depth images are only for one grasping point. For each of the garments, we have 20 to 50 predefined grasping points with respect to the geometric complexity of the garment.

In this chapter, we are focusing on the recognition of deformable object category and pose via depth images as one component of a larger pipeline for manipulating deformable objects such as

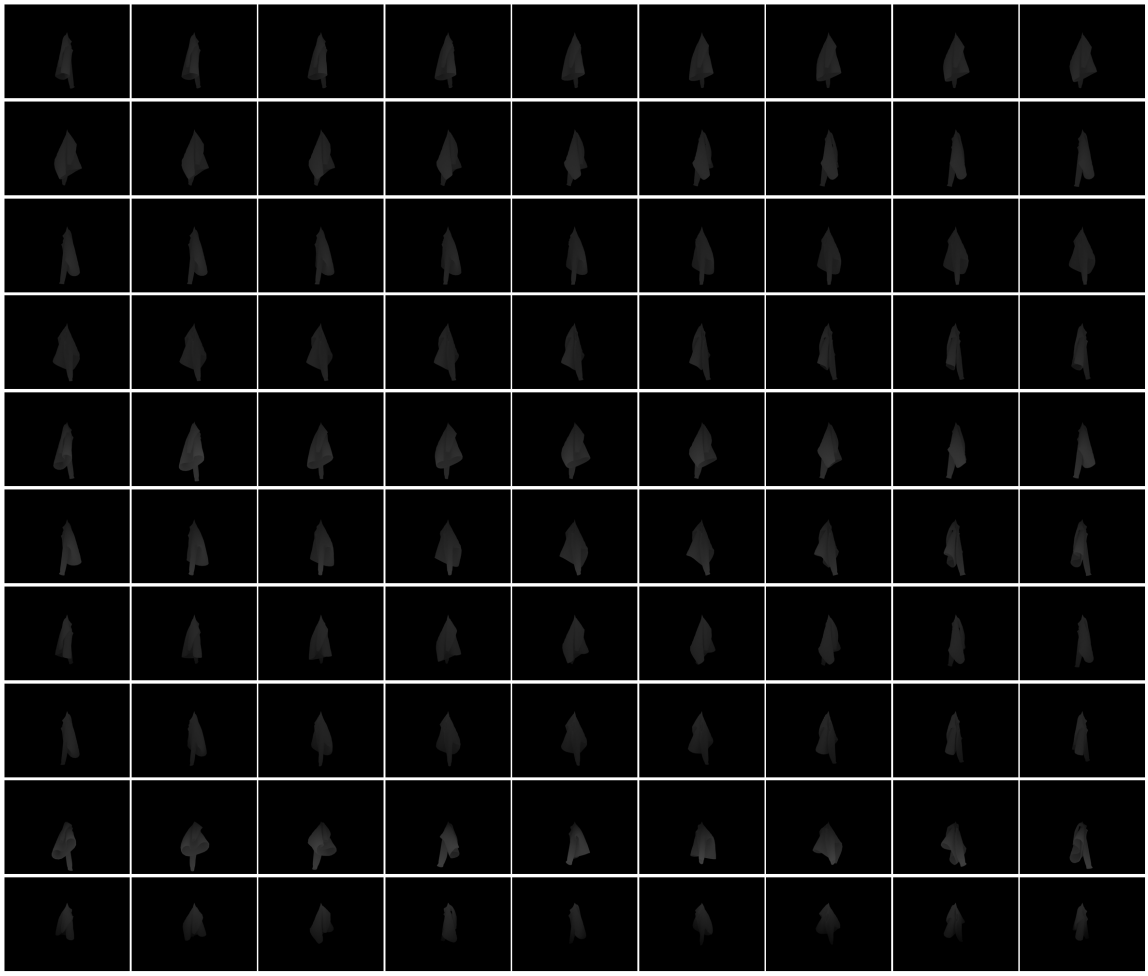


Figure 3.3: 90 depth images captured from the 90-virtual-camera system which is shown in the Figure 3.2.

clothing. The main idea of our method is to use the depth images from the off-line simulation of models of garments in different poses to predict online category and poses. Key contributions of this chapter are:

- A method for estimating deformable object pose using simulation results and dictionary learning via spatial pyramid matching and sparse coding.

- A system that captures 90 depth images from different view points of a target object in $20-50$ different poses. Figure 3.4 shows two garment mesh models and one of their poses hanging by gravity.

- Formulation of the pose recognition problem as a hierarchical image classification task that uses depth images regardless of complicated virtual texture on the object.

- Experimental results in simulation, with real clothing, and also with a physical robot on a variety of deformable objects including sweaters, jeans, and short pants in different sizes. The results show that our approach is robust to estimate the object pose and to facilitate regrasping and folding tasks.

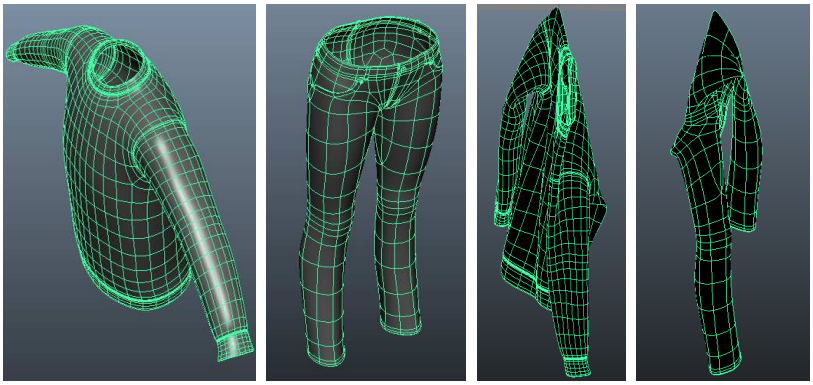


Figure 3.4: (a), (b): Garment mesh models of sweater and jeans rendered in *Maya*. (c), (d): Simulation of hanging under gravity of the garment mesh models, respectively.

3.3 Estimating Poses of Deformable Objects

The goal of building a deformable object database is to help in recognizing the category and the pose of a deformable object. In our research scenario, the robot may randomly grasp and pick up a garment without any prior knowledge. After the grasping, we are interested in identifying the category of the garment first and then the position of the grasping point, which are both very important for further actions such as regrasping and folding. From a technical point of view, given a set of features computed from a vision source, (e.g. a depth camera) we would like to index into the deformable object database via feature space, to identify the pose of a targeted deformable object. The problem can be then formulated as an image classification problem. Given a set of depth images captured from a single object, we are interested in identifying the category and grasping point.

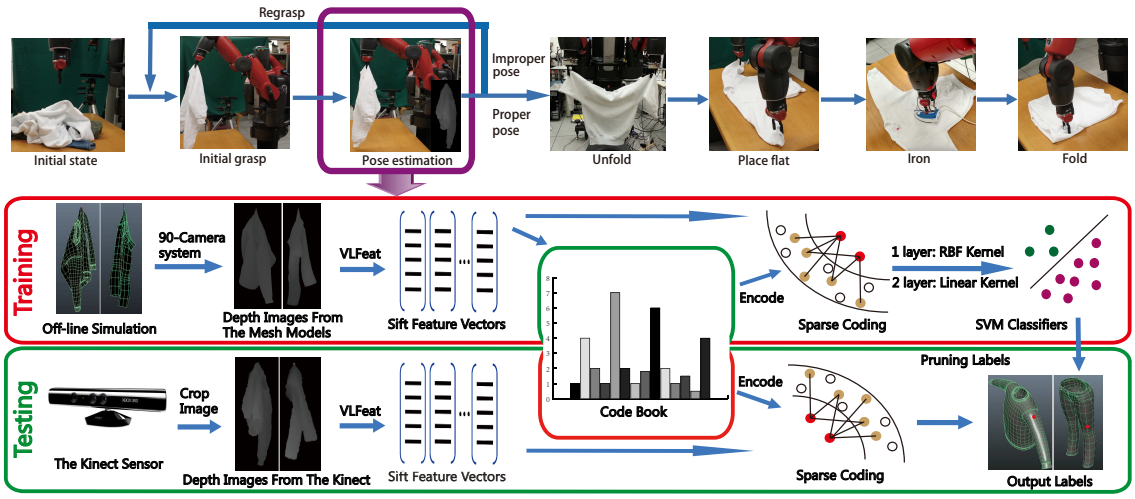


Figure 3.5: Overview of our proposed pipeline for manipulation of deformable objects. TOP ROW: The entire pipeline of dexterous manipulation of deformable objects, which contains five phases. In this chapter, we are focusing on the phase of recognition and identification of deformable object category and pose, as highlighted in purple rectangle. If the recognition and identification are not successful, the robot will regrasp the object and repeat the same process again for recognition and identification. BOTTOM TWO ROWS: In training flow (in red rectangle), we apply off-line simulation of the deformable objects such as mesh models of sweater and jean in different poses. By using sparse coding and dictionary learning on depth images, we build up a codebook for recognition. In testing flow (in green rectangle), we capture depth images by the Kinect sensor and use the precomputed codebook to predict the categories and poses of deformable objects.

3.3.1 Feature Extraction

Our method uses the SIFT descriptor which has some outstanding properties, such as rotation invariance. Unlike the approach of using the descriptor in [Lowe, 2004], we applied Dense SIFT over our depth images using VLFeat [Vedaldi and Fulkerson, 2008]. We also crop the original depth images in terms of the position of the object. This will guarantee that the object always stays in the center of the image, which insures that the spatial pyramid pooling, described below, achieves a better performance. In addition, we discard those sample positions of descriptors that not on the object area for fast computation and encoding.

3.3.2 Generating and Learning Feature Signatures

Sparse Coding

There are two important components in the *Bag of Words* model: dictionary learning and feature quantization. The goal of the BOF model is to find a new feature space that has a better ability to represent the training features. Therefore, how to build a *codebook* is crucial in the BOF model. We define a set of N SIFT descriptor as $\{\mathbf{X}\}^N$, where $\mathbf{X}$ is a sample descriptor in the depth image. The problem is then formulated as the following cost function:

$$\min_{V} \sum_{1}^{N} \min_{i=1...k} \|\mathbf{X}_i - \mathbf{v}_i\|^2 \tag{3.1}$$

where $\mathbf{V} = [\mathbf{v}_{1,...,}\mathbf{v}_k]^{\mathrm{T}}$ is the final k clustering vectors and $\mathbf{v}_{1,...,}\mathbf{v}_k$ are the codewords. We can rewrite the cost function (3.1) into the following matrix factorization:

$$\min_{W,V} \sum_{i=1}^{N} \|\mathbf{X}_i - \mathbf{w}_i\mathbf{V}\|^2 \tag{3.2}$$

where $\mathbf{W} = [\mathbf{w}_{1,...,}\mathbf{w}_N]^{\mathrm{T}}$ is the weight vector for each of the clusters which will be solved. Traditional method usually applies constraints such that $Cardinality(\mathbf{w}) = 1$ (means only one element of $\mathbf{w}$ is nonzero), and $|\mathbf{w}_i| = 1$ which lead to vector quantization method. Here $|\mathbf{w}_i|$ represents l_1 norm of $\mathbf{w}$. This method only allows feature vectors of one sample be assigned to one "word". Sometimes, it is too restrictive that the *codebook* may have a coarse reconstruction of the original feature vectors. In our scenario, it is highly likely that a depth image from the current view point is in between two or more predefined view points. So it may be inaccurate to reconstruct the current sample via only one word in the *codebook*. In order to improve this, we apply sparse coding method. In the work of Lee et al. [Lee *et al.*, 2007], it is stated that if another $L1$ norm was added to the cost function (3.1), the result yields sparsity property, which is called sparse coding. Equation (3.3) shows the sparse coding equation after we add the $L1$ norm term.

$$\min_{\mathbf{W},\mathbf{V}} \sum_{1}^{N} \|\mathbf{X}_i - \mathbf{w}_i\mathbf{V}\|^2 + \lambda |\mathbf{w}_i| \tag{3.3}$$

In sparse coding, there is more than one non-zero number in $\mathbf{w}_i$, meaning the feature vector can be reconstructed via several "words" in the dictionary. In our targeted problem, the pose of the

garment is continuous whereas the training poses are discrete (currently we have defined 90 poses for each grasping point). It is highly likely that the pose of one input garment is in between several training poses. So sparse coding scheme will reduce the errors in reconstruction of the input feature vectors. Since the training task involves large amounts of data, we also employ an online training algorithm that dynamically adjusts the model with less memory consumption (see [Mairal *et al.*, 2009][Mairal *et al.*, 2010] for more details on this method).

Max pooling and Linear SVM

In our recognition pipeline, we are interested in a fast classification speed. After the training feature vectors are quantized, we apply a SVM to learn the model. Though the non-linear SVM is more accurate in most of the cases, it brings a high training cost of $O(n^3)$ and a storage cost of $O(n^2)$. According to Yang's [Yang *et al.*, 2009] work, spatial max-pooling combined with linear-SVM [Fan *et al.*, 2008] for classification yields a better solution for a large data set. After sparse coding on the feature vector, we apply spatial pyramid construction to the feature vector to preserve spatial information. Let $\mathbf{W}'$ be the output weight vector after sparse coding and spatial pyramid construction. The max-pooling function $\mathcal{F}$ is defined on each column of $\mathbf{W}'$. Then the weight vector $\mathbf{r}$ for one depth image can be computed by $\mathcal{F}(\mathbf{W}')$ where $\mathbf{r} = [r_1 ... r_J]$. The function $\mathcal{F}$ computes the max weight among each column that is defined as:

$$r_j = \max\{\left|w'_{1j}\right|, \left|w'_{2j}\right| ... \left|w'_{Nj}\right|\} \tag{3.4}$$

According to the observation in [Yang *et al.*, 2009], spatial max-pooling on sparse coding obtains a high classification accuracy because of less quantization error in sparse coding, robustness in max-pooling to local translation, and sparsity in the image patch. This strategy is only applied to the second layer of the classification which will be described below.

3.3.3 Defining Deformable poses

Our approach is to use a large simulated dataset of the garments that have been picked up and hung by gravity to predict some unknown poses. For every pick up (grasping point), a pose is defined. Specifically, we predefine a set of grasping points on the garment mesh in terms of structural complexity of the garment. At the same time, we can define the region of interest by eliminating those redundant grasping points based on symmetric geometry of the garment. For example, a sweater,

Figure 3.6: LEFT: A real sweater with 50 labels pasted on it which are corresponding to the predefined 50 labels in the figure 3.7. MIDDLE: A real jean with 40 labels pasted on it. RIGHT: A real short pants with 25 labels pasted on it.

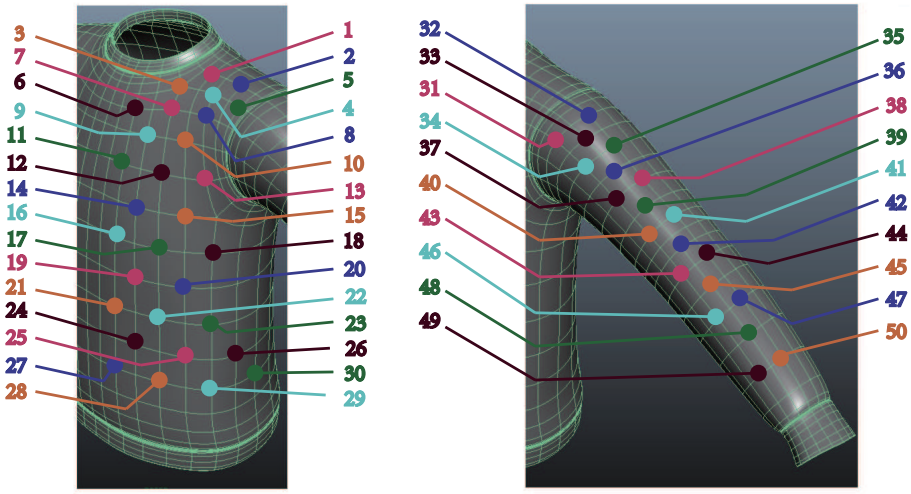


Figure 3.7: 50 predefined grasping points within the region of interest on the sweater mesh model. All the grasping points will be picked up once and simulated individually.

which is symmetric between front and back, left and right so that a reasonable set of grasping points should be within one fourth of the entire surface area. Figure 3.6 shows color coding label pads on real garments within the region of interest.

Considering the sweater model as an example, given a region of interest, we predefine 50 grasping points on the mesh empirically and try to make them uniformly distributed within the region. These points will be treated as "constraints" in the *Maya* simulation, whereas in real experiments,

there are grasping points or hanging points. For each of the predefined points, we simulate the draping effect in *Maya* and a set of 90 depth images are captured as the training data of this particular grasping point. A sample of the predefined points on the sweater model is shown in Figure 3.7. We also apply the same predefined points on the real sweater, jean, and short pants which are shown in Figure 3.6. The real garments we use are very similar to the mesh models we simulated in *Maya*.

Algorithm 1: Label Pruning Algorithm

Input: A predicted label set $\mathbf{\Omega} = \{\mathbf{L_1}, \mathbf{L_2}...\mathbf{L_{n_d}}\}$

Geometry table T contains 3D coordinates of each label

Output: One predicted label $\mathbf{L_p}$ for the current pose

$n = ||\mathbf{\Omega}||$;
$center1 \leftarrow 0, center2 \leftarrow 0$;
while 1 **do**
    $center1 \leftarrow$ mean($\mathbf{\Omega}$);
    $dist\_table \leftarrow$ pdist($center1, \mathbf{\Omega}$);
    $\mathbf{\Omega} \leftarrow \mathbf{\Omega} - \{5farthest\_points\}$;
    $center2 \leftarrow$ mean($\mathbf{\Omega}$);
    if $||center1 - center2|| < \xi$ **then**
        break;
    if $||\mathbf{\Omega}|| < 0.5 \times n$ **then**
        break;
$dist\_table \leftarrow$ pdist $(center2, T)$;
$\mathbf{L_p}$ = find label with $\{min(dist\_table)\}$;
return $\mathbf{L_p}$

3.3.4 Two-layer Classifier

We use a two-layer hierarchical classifier whose structure is shown in Figure 3.8. For both layers, we first extract features and build up the codebook. Then by applying the sparse coding, we encode features using the codebook. Finally, we use the SVM to classify the garment category and pose.

Figure 3.5 also shows the described work flow.

On the first layer (in red shade), each of the input depth images vote for the garment category and the dominant one yields the final category. On the second layer (in green shade), each input depth image will give an output – a grasping point label. For each grasping point, we may have n_d depth images so that we will have n_d predicted labels correspondingly. Here we apply a label pruning algorithm, which iteratively removes predicted labels that are far from the current mean of all labels, and then outputs one label from a set of labels. Details are described in Algorithm 1.

The only difference between the two layers is that on the first layer we applied a Radial Basis Kernel function (RBF) SVM while the Linear Kernel function SVM was used on the second layer. The RBF kernel is more accurate but relatively slow, so we apply it to the first layer where fewer classes exist. The Linear kernel is applied to the second layer which usually has 20 to 50 classes.

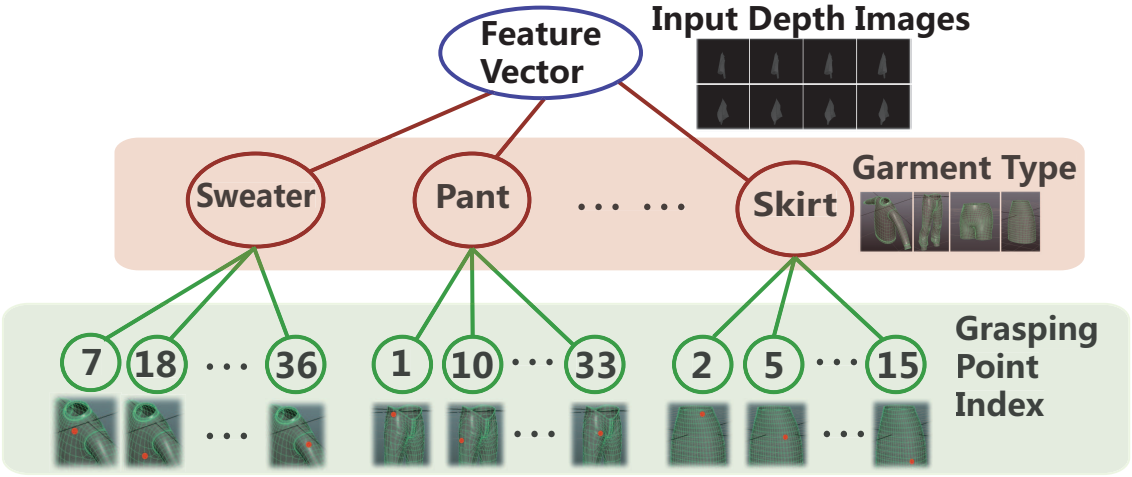


Figure 3.8: Structure of the two-layer classifier. The first layer predicts the categories of the garment candidates using an SVM with an RBF kernel whereas the second layer predicts the grasping point label using an SVM with a linear kernel.

3.4 Experimental Results

3.4.1 Test Using Simulation Data

This test focuses on the performance of the second layer of the classifier using simulation data. We have one data sample for one grasping point label. Therefore, each time we take one entire sample

		Sweater	Jean	Short Pants
Height Group	# of Cam	Rank 1	Rank 1	Rank 1
Group 1	9	6.06	6.12	7.19
Group 2	18	4.93	5.12	5.67
Group 3	36	**4.52**	**4.44**	**5.48**
Group 4	18	4.84	5.06	6.29
Group 5	9	5.38	6.05	7.29
Group 2, 3, 4	72	5.02	5.09	5.81
All	90	5.79	5.73	6.76

Table 3.1: Ranking table of the garment models (sweater, jean, and short pants) using simulation data from *Maya*. The training images are divided into 7 different groups by height and tested individually. We can see that the rank 1 for sweater and jean is between 4 and 5 on average. It means the estimated grasping points is very close to the take-out label according to the distribution of the grasping point labels shown in Figure 3.7. The rank 1 for short pants is a little bit higher which is because the depth images of the short pants are less informative. We can also see that height group 3 always achieves the best recognition rank among all groups.

of a grasping point label out and test it on the classifier which is trained by the rest of the grasping point label samples. If the output labels are close to the take-out label, then the simulation data can be considered continuous and our algorithm should perform well on similar real depth images from a range sensor such as the Kinect.

Taking the sweater model as an example, which has 50 grasping points, each time we select a different target class (a set of depth images from one grasping point) for testing and the rest of the classes (the remaining 49 classes) for training. All depth images can be divided into 5 groups based on their view perspective heights shown in Figure 3.9 as individual training sets, which are labeled from 1 to 5 from top to bottom. We also add group 2+3+4 and all groups to the training sets. In our testing strategy, since we take out the entire testing class, we expect that the estimated grasping point is close to the testing grasping point label. One way to evaluate it is to use the distance ranking as a metric. For example, in Figure 3.7, if we take out the $17th$ grasping point, the $14th$, $15th$, $16th$, $18th$, $19th$, and $20th$ grasping points can be considered as the closest six points in a sorted order by distance to the $17th$ point. If the $14th$ point is identified as the output, the rank is 1.

We apply this strategy to the simulation data which has three categories, each of which has $25-50$ predefined grasping points. The ranks for all camera groups, and several group combinations are shown in Table 3.1. From the table, we can see that the best rank for sweater and jean are between 4 and 5. This is because on average, there are $4-5$ labels with similar distances to the take-out label, which are considered as the closest label sets. The best rank for short pants is a little bit higher because the depth images of its deformable poses are not as informative as those for sweater and jean. In real experiments, we also observed that in terms of camera view points of the garment, selecting depth images from certain camera group(s) as the training data yields better recognition results.

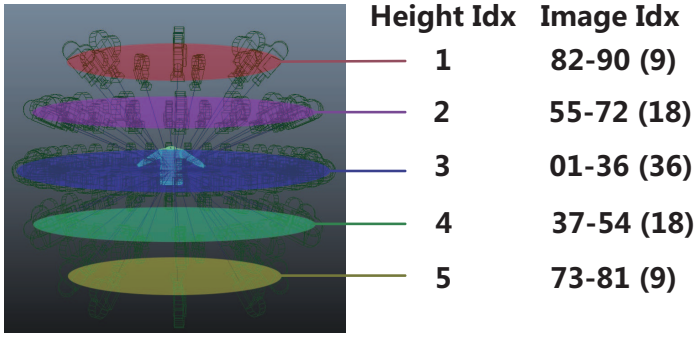


Figure 3.9: 90 cameras are divided into 5 groups based on their view point angles. For each of the groups, the numbers of depth images are shown on the right with index range. For each of the grasping points in simulation, 90 depth images are captured.

	Garment Categories		
Height Group	Sweater	Jean	Short pants
Group 3	75.79%	63.33%	84.71 %
Group $2, 3, 4$	**79.61%**	62.83%	90.76 %
All	73.77%	**72.73%**	**91.40%**

Table 3.2: Accuracy table of the garment models (sweater, jeans, and short pants) using depth images from the Kinect sensor. We can see that using all depth images from simulation for training yields better classification results.

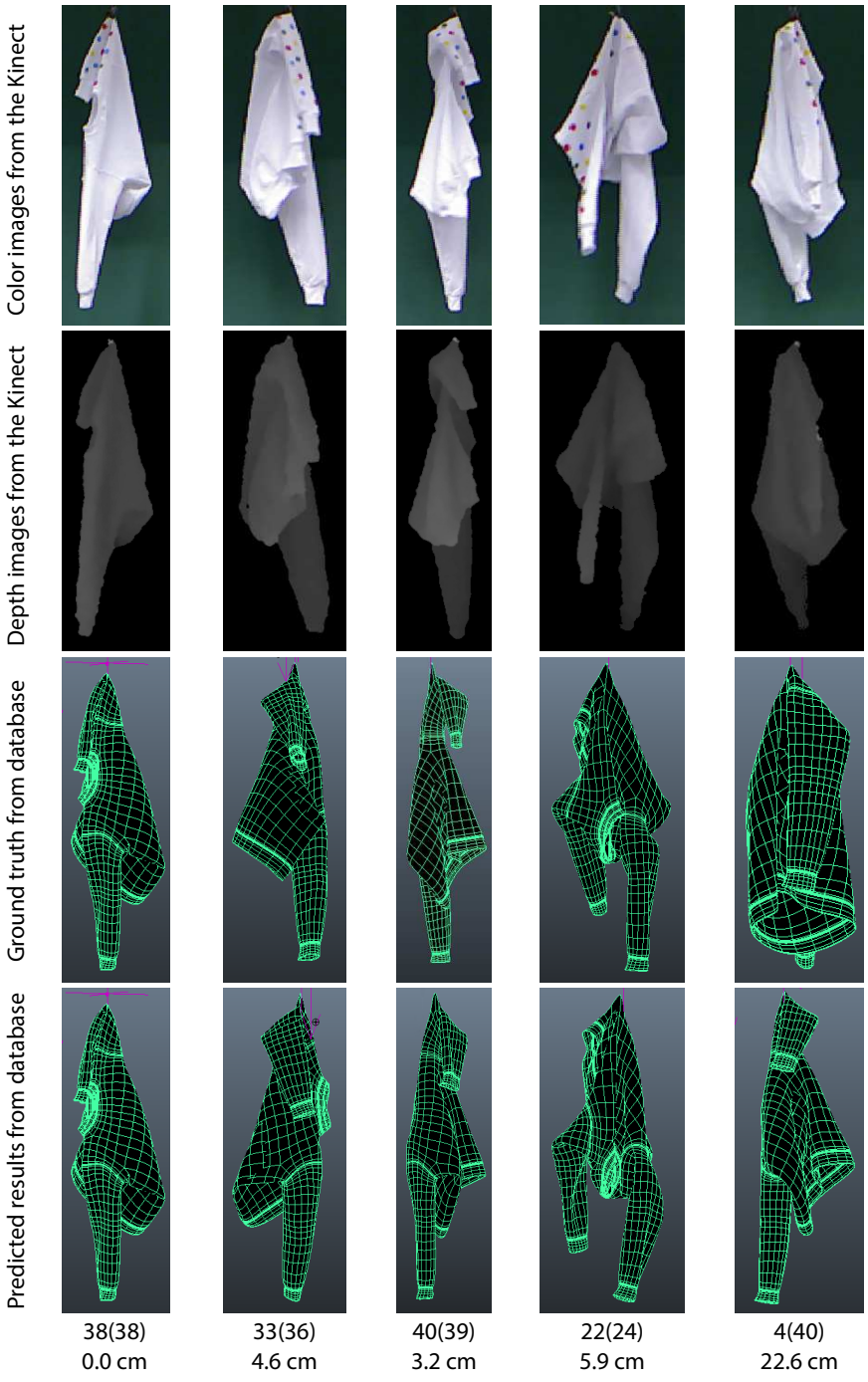


Figure 3.10: Sample results of sweater recognition from the experiments. The top row is the color images captured by the Kinect sensor with different grasping points. The second row is the depth images captured by the Kinect sensor which are in the same pose corresponding to their color images in the first row. The third and fourth rows show the simulation results of garment mesh models in *Maya* hanging by the ground truth grasping points and predicted grasping points, respectively. The two numbers below the fourth row are the ground truth grasping point labels and predicted labels (in parenthesis). The numbers followed by "cm" are the estimated distances between the ground truth and the predicted labels on the garments. The last column is one failure output example of our approach.

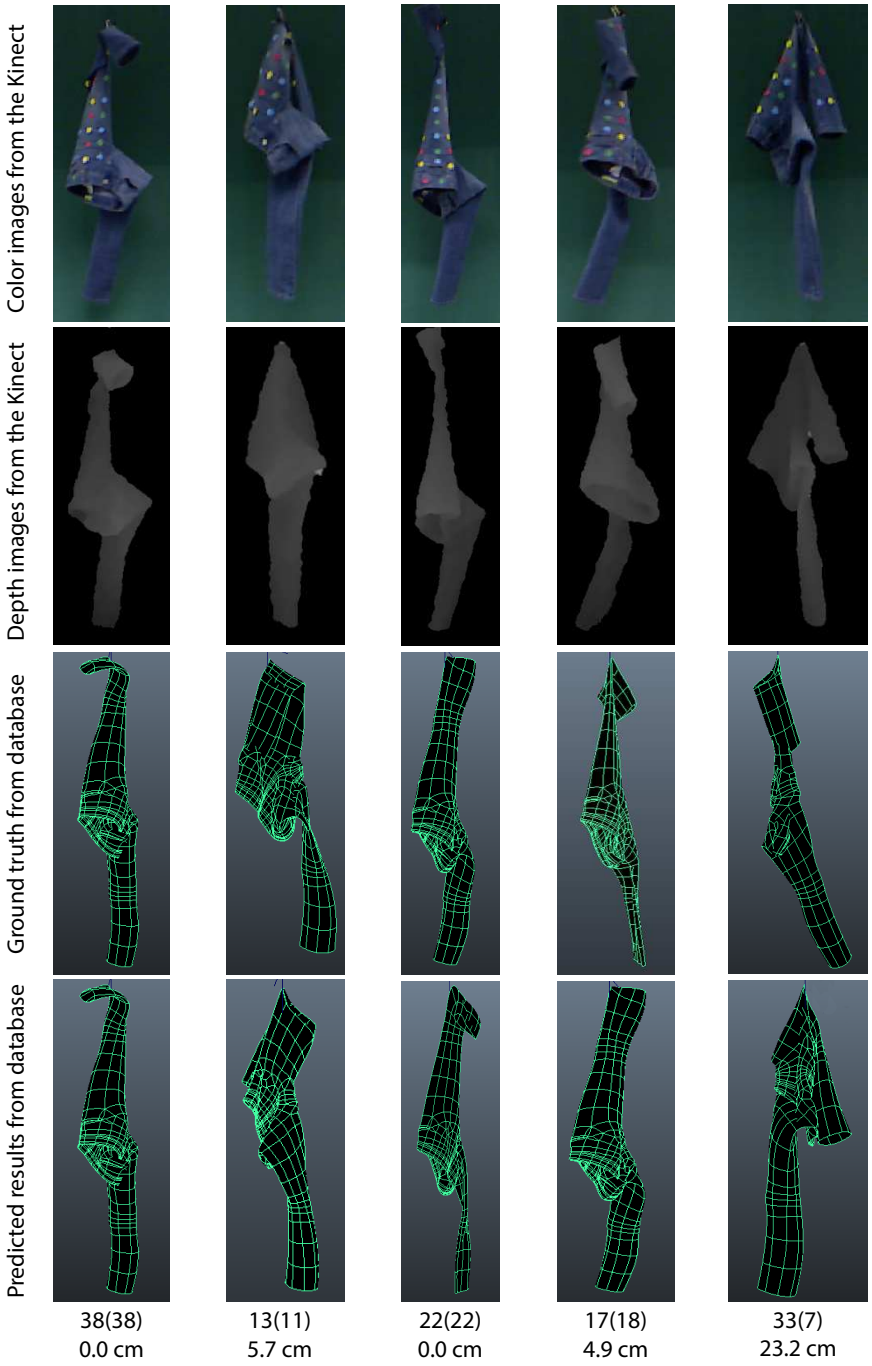
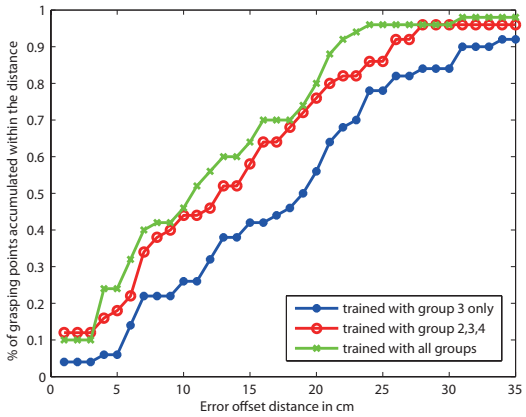
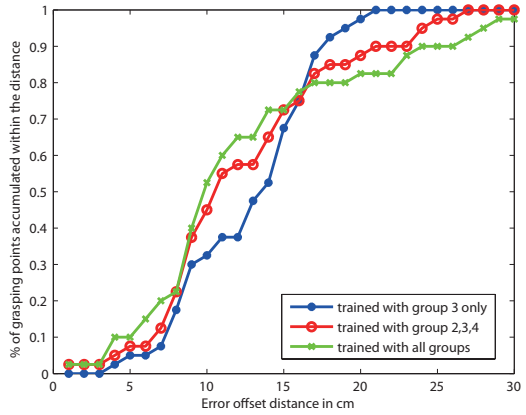


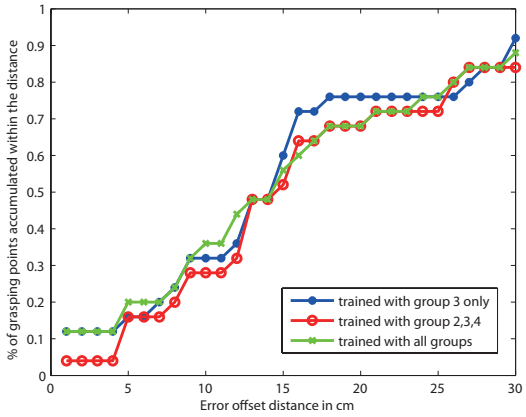
Figure 3.11: Sample results of a pair of jeans recognition from the experiments. The top row is the color images captured by the Kinect sensor with different grasping points. The second row is the depth images captured by the Kinect sensor which are in the same pose corresponding to their color images in the first row. The third and fourth rows show the simulation results of garment mesh models in *Maya* hanging by the ground truth grasping points and predicted grasping points, respectively. The two numbers below the fourth row are the ground truth grasping point labels and predicted labels (in parenthesis). The numbers followed by "cm" are the estimated distances between the ground truth and the predicted labels on the garments. The last column is one failure output example of our approach.



sweater



jean



shortpants

Figure 3.12: Accuracy plot for the garment candidates with different training height groups. The X-Axis is the error offset distance measured in cm. The Y-Axis is the percentage of grasping point accumulated within the distance that corresponding to the X value.

3.4.2 Test Using Depth Images from the Kinect

Our next experiment is to predict the grasping point of a garment from the depth images captured by a real sensor. We use the Microsoft Kinect as the vision sensor to capture depth images at the same height of the garment. We used the same (or very similar) garments to our models in off-line simulation. We manually labeled the garments by using color coding label pads, shown in Figure 3.6, with comparison to the pre-labeled simulated models shown in Figure 3.7. Then for each of the labels (label pads), we hung and rotated the garments about the grasping point vertically and took 150 to 200 depth images with more than 2 rotating circles.

Table 3.2 shows the accuracy of the category classification. (The first layer in Figure 3.8) In this experiment, we treat each grasping point as an individual test group. Each of the depth images in a test group votes for a garment category. If more than 60% of the depth images vote for one category, we take the result as an output judgment. Otherwise, we discard the current grasping point and move to another grasping point. On average, we discard 20% of the grasping points. The results in Table 3.2 are calculated using this strategy. In real experiments, this can be done by regrasping the garment and repeating the whole recognition process. Currently we have three categories which are sweater, jean, and short pant. Among the depth images taken from various poses and view points from a garment, we also notice that there exists a few, that even for human observers, it is very hard to identify the garment category.

Figure 3.10 and Figure 3.11 shows some output results of our algorithm on single depth images. Specifically, the Figure 3.10 is the sweater while the Figure 3.11 is the jean. The top row is the color images captured by the Kinect sensor with different grasping points. Please note that we calibrate the color images to their corresponding depth images so there exists some blurring in the color images. The second row is the depth images, which are captured by the Kinect sensor as well. The depth images shown are after preprocessing such as filtering, etc. The third row shows the simulation results of garment mesh models in *Maya* hanging by the ground truth grasping points with similar view perspectives. The fourth row shows the simulation results of garment mesh models in *Maya* hanging by the predicted grasping points with similar view perspectives. We demonstrate both the ground truth and the predicted results to show that our approach can always identify the grasping point with the same or similar appearance from database. The two numbers below the fourth row are the ground truth and predicted grasping point labels (in parenthesis). For example,

33(36) means the ground truth grasping point label is 33 whereas the predicted grasping point label is 36. The number below followed by "cm" is the actually distance between the two labels on the garments. For example, 33(36) and 4.6 cm indicate that the distance between predefined grasping point label 33 and 36 is 4.6 cm. We can see that the appearance hanging by the predicted grasping points is more similar to the real clothing compared with hanging by the ground truth grasping point.

The last columns of the sweater and jean samples in Figure 3.10 and Figure 3.11 are two failure examples. Comparing the ground truth with the predicted results from database in these two columns, we can observe that the appearance of the predicted results looks more similar to the real garments. It is because the material property of the real garment is different from the property applied in the off-line simulation, which causes varieties in deformation. In future, we plan to apply different material properties to the garment mesh models in the off-line simulation [Umetani *et al.*, 2011].

In Figure 3.12, we apply different training sets (height group 3, height group 2+3+4, and all) to test depth images. The X-coordinate of each point in Figure 3.12 represents the tolerance distance in cm whereas the Y-coordinate represents the accumulated grasping point labels within this distance over the entire garment. The maximum distance between the two grasping points on the sweater, a pair of jeans, and a pair of short pants are 70 cm, 65cm, and 51 cm, respectively. For example, one point (10.7, 0.5) on the green curve (trained with all groups) in the plot of the sweater is interpreted as: if we set 10.7 cm as the tolerance distance, 50% of the predefined grasping point labels are recognized correctly. In our case, there are 50 predefined grasping point labels on the sweater mesh model so 25 labels are considered as correct. We observe that in our experiments, training with all depth images achieves slightly better results. We also observe that within 15 cm tolerance distance, around 70% of grasping points can be identified. We believe this accuracy is sufficient for regrasping and folding tasks.

3.4.3 Implementation on a Robot

We implemented a grasping algorithm on a Staubli arm and a Barrett hand that picked up a real garment and applied our algorithm to recognize its category and pose. We tested our algorithm on real sweater and jeans through 8 times robot trials of picking up and hanging clothing. The maximum distance between the two grasping points on the sweater and the jeans are 70 cm and

65cm. Table 3.3 shows the results of the the two-layer classifier in the robot experiments. We can see that our algorithm can always predict the grasping point within a short distance from the ground truth. Figure 3.13 shows snapshots of the process of the recognition pipeline using our approach which contains initial state, grasping and picking up, and recognition from different view points. A video of our experimental results is available at: http://www.cs.columbia.edu/~yli/ICRA2014

	sweater				jean			
#	1	2	3	4	5	6	7	8
Category	S	S	S	S	J	J	J	J
Distance (cm)	3.2	15.7	9.1	3.0	7.6	7.4	15.1	10.8

Table 3.3: Results of the robot experiments. Totally we have 8 trials. First we classified the category for sweater (S) and jean (J). Then we provide the distance in cm between the predicted grasping point and the ground truth.

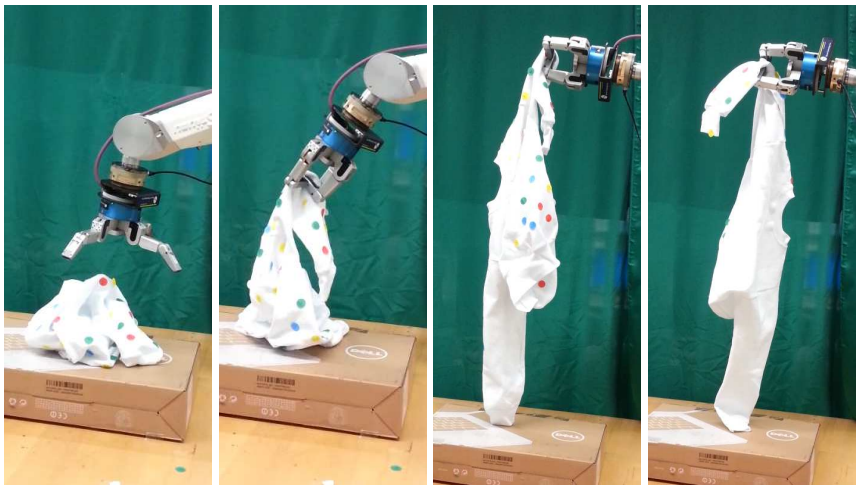


Figure 3.13: A Staubli arm grasps a real garment, picks it up, hangs it under gravity, rotate it, and recognizes its category and pose using a Barrett hand. From left to right: initial state, grasping and picking up, recognizing from one view point, rotating and recognizing from another view point.

3.5 Summary

The entire pipeline of dexterous manipulation of deformable objects includes grasping, recognition, regrasping, placing flat, and folding deformable clothing objects. In this chapter, we focus on the recognition. We have shown a framework for recognizing the categories and the poses (grasping points) of a deformable object. We first built up a deformable object database with a set of depth images from off-line simulation as the training data. Then sparse coding was applied to better quantize feature vectors and construct a codebook. We also proposed a two-layer classifier for our classification task. We then tested the performance of the classifier using the off-line simulation data. Meanwhile, we took real depth images via the Microsoft Kinect for testing as well. Both testing results have shown that our proposed framework worked well for estimating the categories and the pose (grasping point) of the deformable object.

Chapter 4

Recognition via Volumetric Approach

This chapter discusses a new recognition method via a volumetric approach, which is an improvement of the recognition via depth perception in Chapter 3.2 both in accuracy and speed. More specifically, our method can achieve real-time category and pose recognition with more accurate prediction of grasping point locations. Figure 5.1 shows a Baxter robot grasping a garment and predicting the grasping location (e.g. 2cm left of the collar). With this information, the robot is then able to proceed to following tasks such as regrasping and folding. The whole pipeline of garment folding is shown in Figure 1.1, whereas our work in this chapter is focusing on pose estimation using volumetric approach. The main idea of our method is to first accurately reconstruct a 3D mesh model from a low-cost depth sensor, and then compute the similarity between the reconstructed model and the models simulated offline to predict the pose of the object. Key contributions of this chapter are:

- A real-time approach to reconstruct a smooth 3D model of a moving (e.g. rotating) deformable object from noisy background without user interaction
- Formulation of the pose recognition problem as a real-time 3D shape matching task with a learned distance metric
- An automatic pipeline for building an offline database of deformable objects using a simulation engine that can be used for efficient data-driven pose recognition
- Experimental results with many different garments that show improved accuracy over our

previous method [Li *et al.*, 2014a], and orders of magnitude speed up yielding real-time performance for the pose estimation task

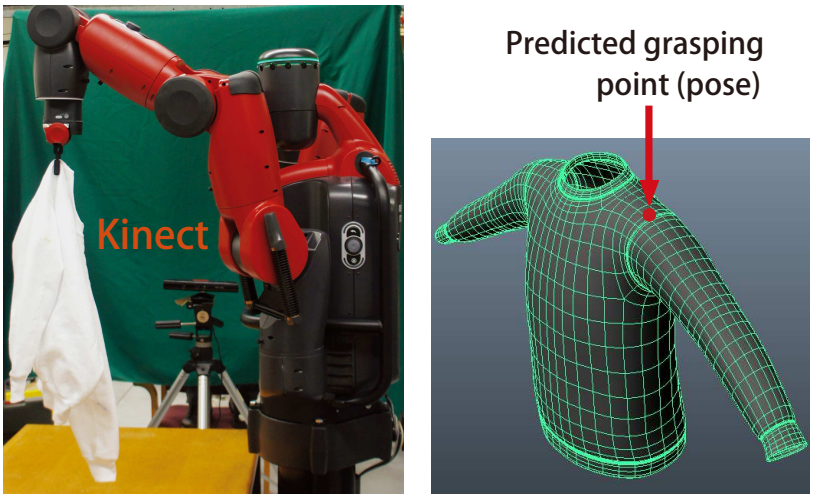


Figure 4.1: Our application scenario: a Baxter robot grasps a sweater, and a Kinect captures depth images to recognize the pose of the sweater. The recognition result is shown on the right.

4.1 Method

Our method consists of two stages, the offline model simulation stage and the online recognition stage. In the offline model simulation stage, we use a physics engine to simulate the stationary state of the mesh models of different types of garments in different poses. In the online recognition stage, we use a Kinect sensor to capture many depth images of different view points of the garment by rotating it as it hangs from a robotic arm. We then reconstruct a smooth 3D model from the depth input, extract compact 3D features from it, and finally match against the offline model database to recognize its pose. Figure 4.2 shows the framework of our method, which will be introduced in the subsequent subsections.

4.1.1 Model Simulation

Pose estimation of a deformable object such as garment is challenging especially because enormous number of possible deformation configurations. However, when an object is grasped by a robotic

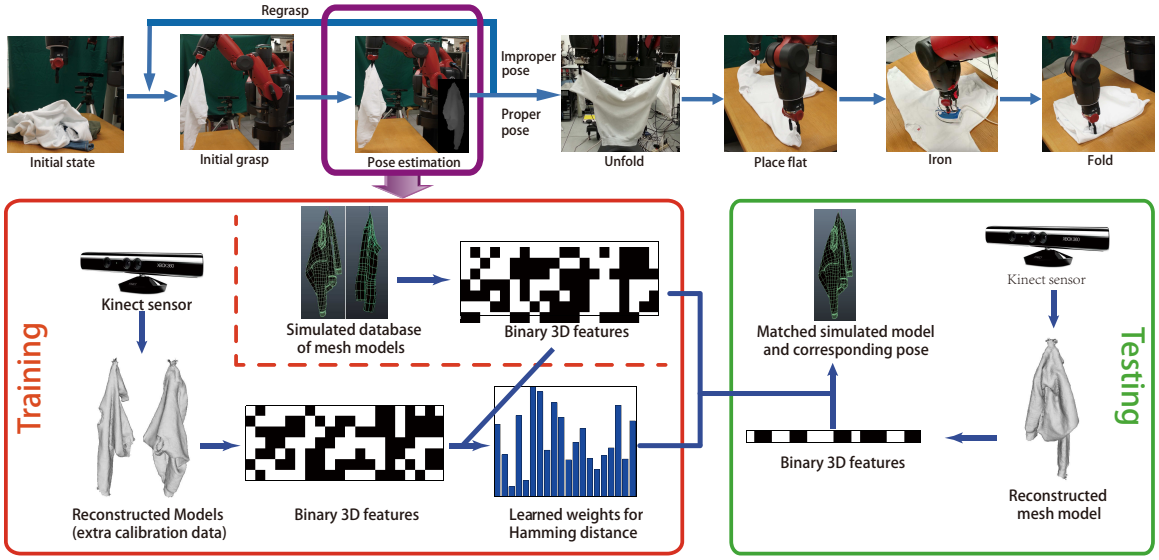


Figure 4.2: Overview of our proposed pipeline for pose estimation of deformable objects. TOP ROW: The entire pipeline of dexterous manipulation of deformable objects. In this chapter, we are focusing on the phase of pose estimation, as highlighted in the purple rectangle. If the recognition is not successful or the pose is improper for the following manipulation, the robot will regrasp the object and repeat the step of pose estimation. BOTTOM ROW: In the offline training stage (the red rectangle), we simulate mesh models of different types of garments in different poses, and learn a weighted Hamming distance from additional calibrated data collected from the Kinect. In the online testing stage (the green rectangle), we reconstruct a 3D model from the depth input, find the nearest neighbor from the simulated database with the learned distance metric, and then adopt the pose of the matched model as the output.

arm, the pattern of the deformation is usually limited. Therefore, we propose to reduce the deformation complexity by first simulating a large database of various collections of garments with different grasping points, and then match the input model against the database. More specifically, for each input mesh model, we first select a set of points from the vertices of the model, and then use a physics engine to compute the mesh model of the garment in the stationary state as if a robotic arm were grasping at each selected point.

To select these grasping points, we first "unwrap" the mesh of the garment to a planar UV map, and then perform uniform sampling on it, as Figure 4.3 shows. The intuition behind is to obtain

a piecewise linear mapping (rotating and minimal stretching in our case) on the vertices such that the result planar faces can preserve the size of the garment, with the final goal to make uniform sampling on the 2D map result in uniformly distributed points in 3D. We use the Global/Local Mapping proposed in [Liu *et al.*, 2008].

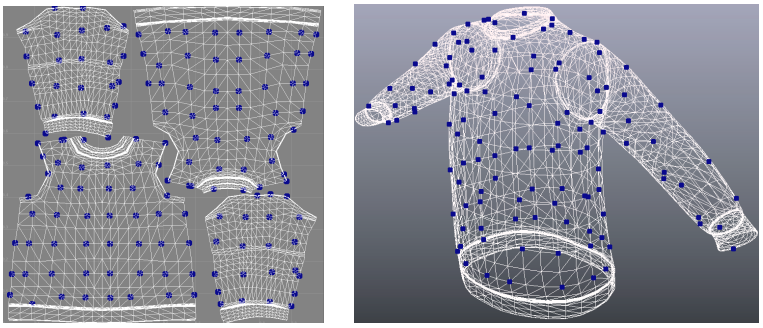


Figure 4.3: An example of generating the set of grasping points for offline simulation. LEFT: The UV map of a sweater mesh model. Grasping points (the blue dots) are selected by finding the closest vertices to the uniformly sampled points. RIGHT: The original sweater model with the selected grasping points mapped back.

After the UV mapping step, we do uniform sampling (in terms of physical size) on the mapped plane. The grasping points are selected as the closest vertices to the sampled points, with one example shown in Figure 4.3. From the figure, we can see the sampled grasping points are generally uniformly distributed. We employ a similar physics simulation method described in our previous work [Li *et al.*, 2014a], with the difference that the final outputs are mesh models instead of rendered depth maps. This simulation stage ends up with a set of mesh models with different garment types, material properties, and grasping points, which will be matched against a reconstructed model from a Kinect sensor.

4.1.2 3D Reconstruction

Given the model database described above, we now need to generate depth images and match against the database. Direct recognition from depth images suffers from the problems of self-occlusion and sensor noise. This naturally leads to our new method of first building a smooth 3D model from the noisy input, and then performing recognition in 3D. However, how to do such reconstruction is still an open problem. Although there are existing approaches of obtaining high-quality models from

noisy depth inputs such as KinectFusion [Newcombe *et al.*, 2011], which requires the scene to be static. In our data collection settings, the target garment is being rotated by a robotic arm, which invalidates the KinectFusion's assumptions. We propose to solve this problem by first segmenting out the garment from its background, and then invoke KinectFusion to obtain a smooth 3D model, assuming that the rotation is slow and steady enough such that the garment will not deform in the process. This is because we only rotate the last joint of the Baxter's arm.

Segmentation. Before diving into the reconstruction algorithm, let us first define some notation. Given the intrinsic matrix F_d of the depth camera and the ith depth image I_i, we are able to compute the 3D coordinates of all the pixels in the camera coordinate system with $\begin{bmatrix} x_{ci} & y_{ci} & z_{ci} \end{bmatrix}^T = F^{-1} d_i \begin{bmatrix} u_i & v_i & 1 \end{bmatrix}^T$, in which (u_i, v_i) is the coordinate of a pixel in I_i, with d_i as the corresponding depth, and (x_{ci}, y_{ci}, z_{ci}) is the corresponding 3D coordinate in the camera coordinate system.

Our segmentation is then performed in the 3D space. We ask the user to specify a 2D bounding box on the depth image $(x_{\min}, x_{\max}, y_{\min}, y_{\max})$ with a rough estimation of the depth of the garment $(z_{\min}, z_{\max})$. Given that the data collection environment is reasonably constrained, we find even one predefined bounding box works well. Then we adopt all the pixels having their 3D coordinates within the bounding box as the foreground, resulting in a series of masked depth images $\{I_i\}$ and their corresponding 3D points, which will be fed into the reconstruction module.

The 3D reconstruction is done by feeding the masked depth images $\{I_i\}$ into KinectFusion, while the unrelated surroundings are eliminated now, leaving the scene to reconstruct as static. This process can be done in real time. In addition to a smooth mesh, the KinectFusion library also generates a Signed Distance Function (SDF) mapping. The SDF is defined on any 3D point (x, y, z). It has the property that it is negative when the point is within the surface of the scanned object, positive when the point is outside a surface, and zero when it is on the surface. We will use this function to efficiently compute our 3D features in the next subsection.

4.1.3 Feature Extraction

Inspired by 3D Shape Context, we design a binary feature to describe the 3D models. In our method, the features are defined on a cylindrical coordinate system fit to the hanging garment as opposed to traditional 3D Shape Context which uses a spherical coordinate system [Frome *et al.*, 2004].

For each layer, as shown in Figure 4.4 top-right, we *uniformly* divide the world space into $(R$

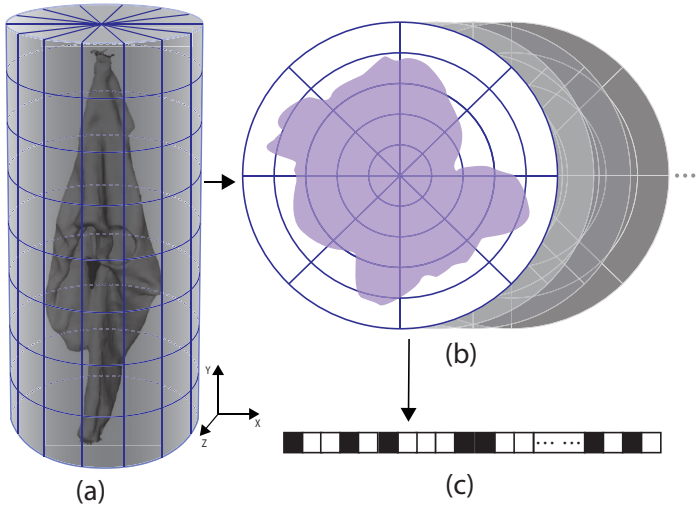


Figure 4.4: Feature extraction from a reconstructed mesh model. (a) indicates that a bounding cylinder of a garment is cut into several layers. (b) shows a set of layers (sections). For each layer, we divide it into cells via rings and sectors. (c) shows a binary feature vector collected from each cell. Details are described in section 4.1.3.

rings) $\times$ (Φ sectors) in a polar coordinate system, with the largest ring covering the largest radius among all the layers. The center of the polar coordinate system is determined as the mean of all the points in the highest layer, which usually contains the robot grasper. Note we do a uniform division instead of logarithm division of r as Shape Context does. The reason why Shape Context uses logarithm division of r is that the cells farther from the center are less important, which is not the case in our settings. For each layer, instead of doing a point count as in the original Shape Context method, we check the Signed Distance Function (SDF) of the voxel which the center of the polar cell belongs to, and fill one (1) in the cell if the SDF is zero or negative (i.e. the cell is inside the voxel), otherwise zero (0). Finally, all the binary numbers in each cell are collected in a order (e.g. with ϕ increasing and then r increasing), and concatenated as the final feature vector.

The insight behind this design is, to improve the robustness against local surface disturbance due to friction, we include the 3D voxels *inside* the surface in the features. Note we do not need to do the time-consuming classification (e.g. ray tracing) to determine whether each cell is inside the surface, but only need to look up their SDFs, thus dramatically speeding up the feature extraction.

Matching Scheme. Similar to Shape Context, when matching against two shapes, we concep-

Algorithm 2: Feature extraction for pose estimation of deformable objects

Input: Vertices of the mesh model $\Omega = \{v_i\}$, precomputed SDF(x, y, z), Parameters: N = #layers, R = #rings, Φ = #sectors

Output: Corresponding feature vector $\mathbf{x} \in \mathbb{B}^{lrs}$

$\mathbf{x} = \mathbf{0} \in \mathbb{B}^{lrs}$;

Divide mesh Ω into l layers $\Omega_1, \Omega_2 \cdots$ in a top-down manner;

Origin = Mean$(\Omega_{1x}, \Omega_{1z})$;

[$\mathbf{r}, \boldsymbol{\phi}$] = Polar (Origin, Ω_x, Ω_z) ;

$r_m = \max \mathbf{r}$;

for *each layer* Ω_i **do**

  for *each cell (ring, sector)* $\in \Omega_i$ **do**

    $(x, y, z) =$ center of the cell ;

    if *SDF*$(x, y, z) \leq 0$ **then**

      $\mathbf{x}\left[\frac{rR\Phi}{r_m} + \frac{\phi\Phi}{2\pi}\right] = 1$;

    else

      $\mathbf{x}\left[\frac{rR\Phi}{r_m} + \frac{\phi\Phi}{2\pi}\right] = 0$;

return $\mathbf{x}$.

tually rotate one of them and adopt the minimum distance as the matching cost, to provide rotation invariance. That is,

$$\text{Distance}(\mathbf{x}_1, \mathbf{x}_2) = \min_i \|R_i \mathbf{x}_1 \oplus \mathbf{x}_2\|_1, \tag{4.1}$$

in which $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{B}^{\Phi RN}$ are the features to be matched ($\mathbb{B}$ is the binary set $\{0, 1\}$), $\oplus$ is the binary XOR operation, and R_i is the transform matrix to rotate the feature of each layer by $2\pi/\Phi$. Recall that both features to be matched are compact binary codes. Thus such conceptual rotation as well as hamming distance computation can be efficiently implemented by integer shifting and XOR operations, such matching is even faster than the Euclidean Distance given reasonable Φs (e.g. $\Phi = 10$). A complete illustration of the feature extraction algorithm can be found in Algorithm 2.

4.1.4 Domain Adaptation

Now we have a feature vector representation for each model in the simulated database and for the query. A natural idea is to find the Nearest Neighbor (NN) of the query in the database and transfer the metadata such as category and pose from the NN to the query. But a naive NN algorithm with Euclidean distance may not work here because even for the same garment and the same grasping point by the robot, the way it deforms may still be slightly different due to friction. This requires a solution in the matching stage, especially given that it is impractical to simulate every object with all the possible materials. Therefore essentially we are doing cross-domain retrieval, which generally requires a "calibration" step to adapt the knowledge from one domain (simulated models) to another (reconstructed models).

Weighted Hamming Distance. Similar with the distance calibration in [Wang *et al.*, 2013], we use a *learned* distance metric to improve the NN accuracy, i.e.

$$\text{BestMatch}_{\mathbf{w}}(\mathbf{q}) = \arg\min_{i} \mathbf{w}^T \left(\hat{\mathbf{x}}_i \oplus \mathbf{q}\right), \tag{4.2}$$

in which $\mathbf{q}$ is the feature vector of the query, i is the index of models in the simulated database, and $\oplus$ is the binary XOR operation. $\hat{\mathbf{x}}_i = \hat{R}_i \mathbf{x}_i$ indicates the feature vector of the ith model, with $\hat{R}_i$ as the optimal R in Equation 4.1.

The insight here is that we wish to grant our distance metric more robustness against material properties by assigning larger weights to the regions invariant to the material differences (this amplifies the features that are more intrinsic for the recognition task).

Distance Metric Learning. To robustly learn the weighted Hamming distance, we use an extra set of mesh models collected from Kinect as *calibration data*. The collection settings are the same as described in Section 4.1.2 and only a small amount of calibration data is needed for each category (e.g. 5 models in 28 poses for sweater model). To determine the weight vector $\mathbf{w}$, we then formulate the learning process as an optimization problem of minimizing the empirical error with a

large-margin regularizer:

$$\begin{aligned} &\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_j \xi_j \\ &\text{s.t. } \mathbf{w}^T(\hat{\mathbf{x}}_i \oplus \mathbf{q}_j) < \mathbf{w}^T(\hat{\mathbf{x}}_k \oplus \mathbf{q}_j) + \xi_j, \\ &\quad \forall j, \forall y_i = l_j, y_k \neq l_j, \\ &\quad \xi_j \geq 0, \end{aligned} \tag{4.3}$$

in which $\hat{\mathbf{x}}_i$ is the orientation-calibrated feature of the ith model (from the database), with y_i as the corresponding ground truth label (i.e. the index of the pose). $\mathbf{q}_j$ is the extracted feature of the jth training model (from Kinect), with l_i as the ground truth label. We wish to minimize $\sum_i \xi_i$, which indicates how many wrong results the learned metric $\mathbf{w}$ gives, with a quadratic regularizer. C controls how much penalty is given to wrong predictions.

This is a non-convex and even non-differentiable problem. Therefore we employ the RankSVM [Joachims, 2002] to obtain an approximate solution using the cutting-plane method, which iteratively refines an objective function by means of linear inequalities.

Knowledge Transfer. Given the learned $\mathbf{w}$, in the testing stage, we then use Equation 4.2 to obtain the nearest neighbor of the query model. We directly adopt the grasping point of the nearest neighbor, which is known from the simulation process, as the final prediction.

4.2 Experimental Results

We used a series of experiments to demonstrate the effectiveness of the proposed method and justify the components. We tested our method on a dataset of various kinds of garments collected from practical settings, and then quantitatively compared the result with our previous work for garment pose estimation described in [Li *et al.*, 2014a]. To evaluate our results, the geodesic distance on the garment between the predicted grasping point and the ground truth is computed, together with the running time. Experimental results demonstrate that our method is able to achieve both higher accuracy and orders of magnitude speed-up. Here we assume the category of the garment is known beforehand.

A video of our experimental results is available at: http://www.cs.columbia.edu/~yli/laundry_robot

4.2.1 Data Acquisition

We collect a dataset for general evaluation of pose recognition of deformable objects based on depth image as inputs. The dataset consists of three parts, a training set, a testing data set, and a calibration set. The training set is the simulated mesh models of different types of garments in different poses, as introduced in Section 4.1.1. We bought 3 commercial-quality mesh models – sweaters, pants and shorts, and simulate each with $80 - 120$ grasping points (different types of garments have different number of grasping points depending on surface area and model complexity). Since all of our garment candidates are symmetric in front and back, left and right, we only adopt those grasping points on one fourth of surface over the whole garment to remove duplicates, ending up with 68 grasping points, each with a corresponding simulated mesh models.

To collect the testing set, we use a Baxter robot, which is equipped with two arms with seven degrees of freedom. A Kinect sensor is mounted on a horizontal platform at height of 1.2 meters to capture the depth images, as shown in Figure 5.1. With this setting, we collect data at the same grasping points of the training set, and then use our 3D reconstruction algorithm as introduced in Section 4.1.2 to obtain their mesh models. For each grasping point of each garment, the robot rotates the garment 360 degrees in about 10 seconds while the Kinect captures at 30fps, which gives us around 300 depth images for each garment/pose. Altogether we have 68 grasping points over the three garments (a sweater, a pair of jeans, and a shorts). This results in a test set of 68 mesh models, with their raw depth images.

Given we also need to learn/calibrate a distance metric from extra data from Kinect, we collect an extra small amount of data with the same settings as the calibration data, only collecting 5 poses for each garment. A weight vector $\mathbf{w}$ is then learned from this calibration data for each type of garment as introduced in Section 4.1.4.

4.2.2 Qualitative Evaluation

We demonstrate some of the recognition results in Figure 4.5 and Figure 4.6 in the order of color image, depth image, reconstructed model, predicted model, ground truth model, and predicted grasping point (red) vs. ground truth grasping point (yellow) on the garment. From the figure, we can first see that our 3D reconstruction is able to provide us with good-quality models for a fixed camera capturing a dynamic scene. And our shape retrieval scheme with learned distance metrics is also

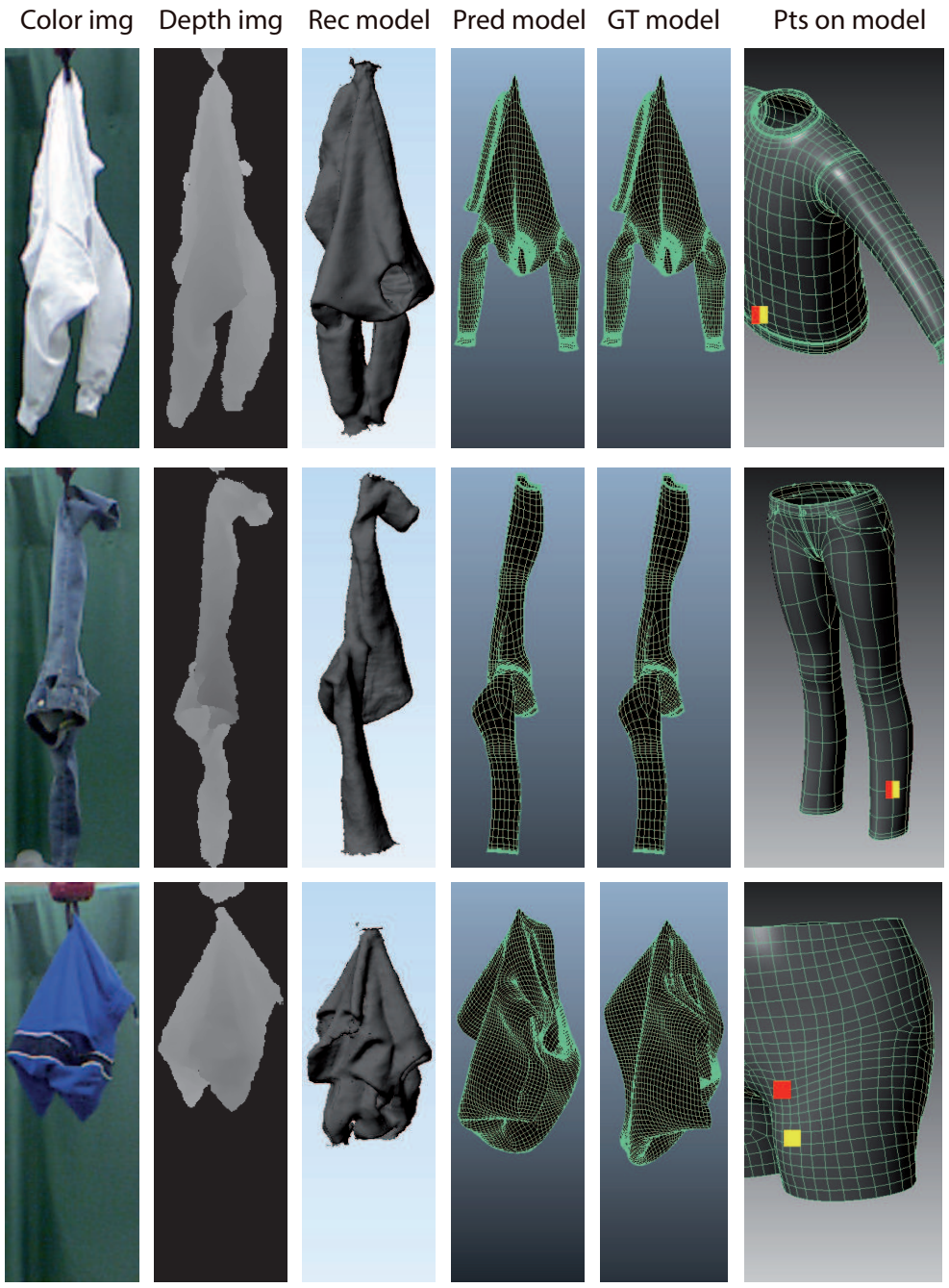


Figure 4.5: Visual examples of the pose recognition result of our method. The garment is picked up via a griper of the Baxter robot. From left to right, each example shows the color image, input depth image, reconstructed model, matched simulated model, ground truth simulated model, and the predicted grasping points (red) marked on the model with the ground truth (yellow). (Best viewed in color)

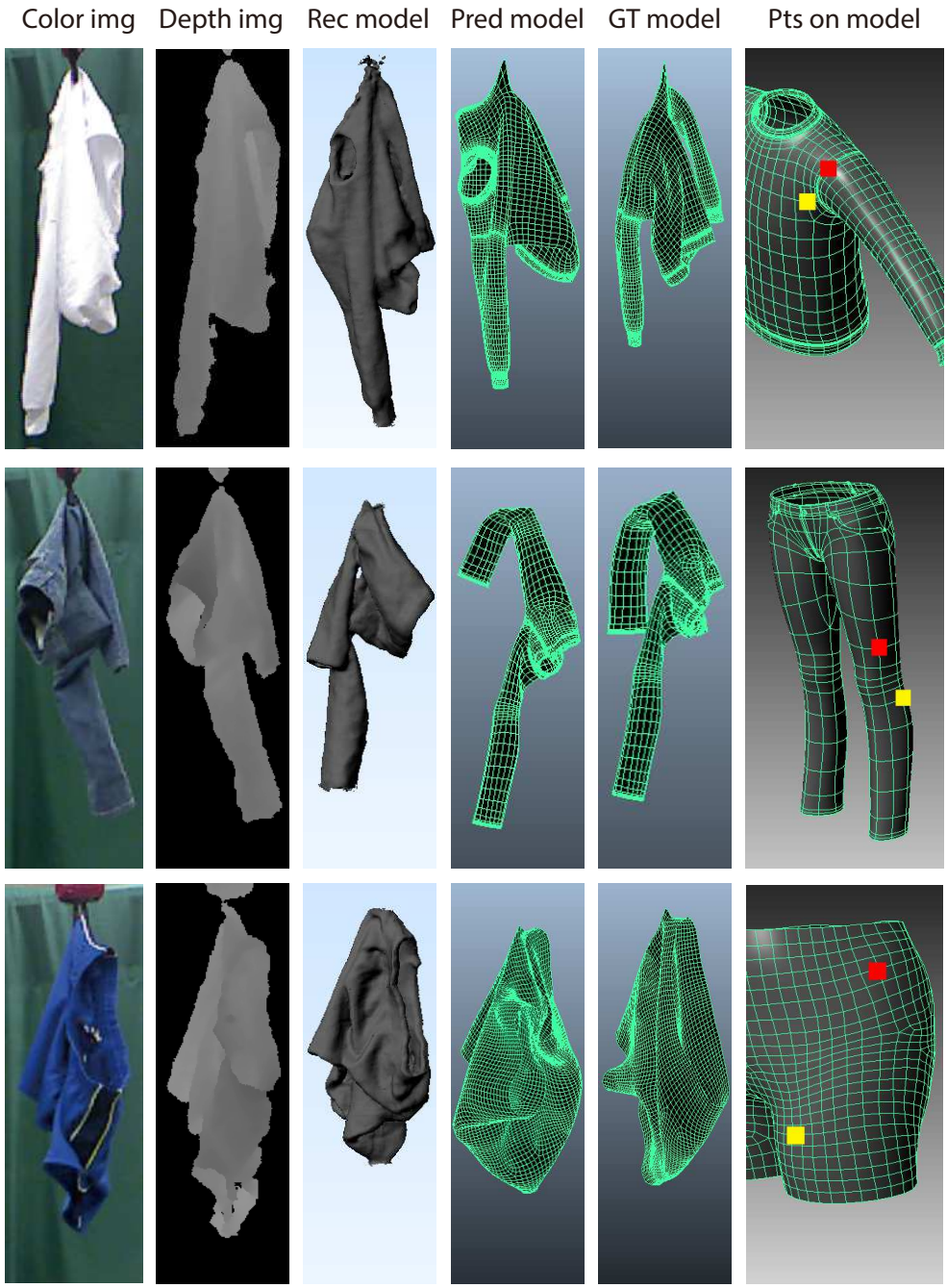


Figure 4.6: More visual examples of the pose recognition result of our method. The garment is picked up via a griper of the Baxter robot. From left to right, each example shows the color image, input depth image, reconstructed model, matched simulated model, ground truth simulated model, and the predicted grasping points (red) marked on the model with the ground truth (yellow). The example shown in the bottom shown here is considered as a failure example, which may be because of the uninformative deformation shape. (Best viewed in color)

able to provide reasonable matches for the grasping points. Note that our method is able to output a mesh model of the target garment, which is critical for the subsequent operations such as path planning and object manipulation.

4.2.3 Quantitative Evaluation

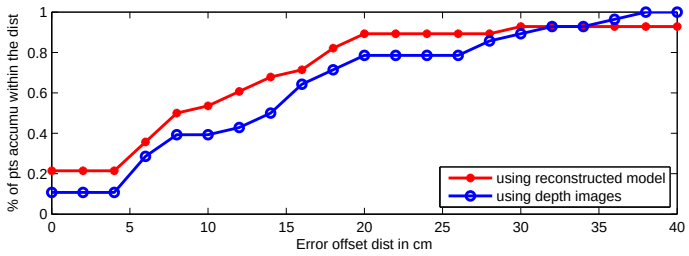
We first introduce some implementation details of our method, and then provide quantitative evaluations.

Implementation Details. In the 3D reconstruction, we set $X = 384$, $Y = Z = 768$ voxels and the resolution of the voxels as 384 voxels per meter to obtain a trade-off between resolution and robustness against sensor noise. In the feature extraction, our implementation adopts $R = 16, \Phi = 16, N = 16$ in the feature extraction as an empirically good configuration. That is, each mesh model gives a $16 \times 16 \times 16 = 4096$ dimensional binary feature. We set the penalty $C = 10$ in Equation 4.3.

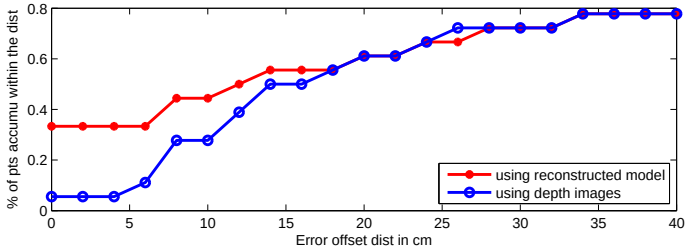
Geodesic Error. For each input garment, we compute the geodesic distance of the predicted point and the ground truth, which we will refer as *Geodesic Error* in the following text, for evaluation. The distribution and the mean of the Geodesic Error are used as the evaluation protocol, and compared our method with our previous method. Since our previous method uses depth images as input, for a fair comparison, we feed all the depth images to our previous algorithm for each pose of each garment.

A comparison of the distribution of the Geodesic Error is plotted in Figure 4.7. The total grasping points for sweater, jeans, and shorts are 28, 20, 20, respectively. We can clearly see that our method outperforms our previous method [Li *et al.*, 2014a] in all the different garment types. Our method benefits from the 3D reconstruction step, which reduces the sensor noise and integrates the information of each frame to a comprehensive model and thus leads to better decisions. Among three types of garments, recognition of shorts is not as accurate as the other two. One possible reason is that many of the shapes from different grasping points look very similar. Even for human observers, it is hard to distinguish them.

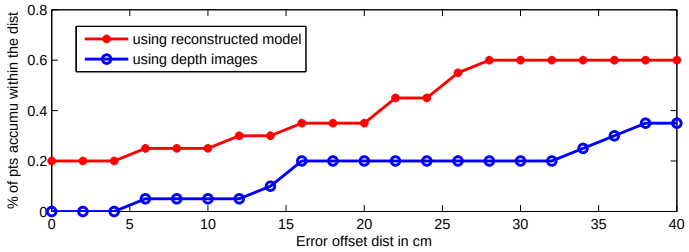
To prove that the domain adaptation is a necessary step in our proposed method, we also test the Geodesic Error of our method without domain adaptation. The mean of the Geodesic errors of different method on different garments is illustrated in Table 4.2. We can see that our method without



Sweater



Jeans



Shorts

Figure 4.7: Quantitative comparison of the proposed method (using reconstructed model) and our previous method (using individual depth images). The x axis is the Geodesic Error, and the y axis is the percentage of the input grasping points which give Geodesic Error smaller than the corresponding x. The top row shows the result of a sweater as input, with maximum distance between any two grasping points as 75 cm. The middle and bottom rows show a pair of jeans and shorts, with maximum distance between any two grasping points as 65 cm and 51 cm respectively.

the domain adaptation cannot beat our previous work, which verifies our motivation of introducing the cross-domain learning. When combined with the learned distance metric, our method is able to achieve lower Geodesic Error than results from [Li *et al.*, 2014a].

Running Time. In addition, we also compare the processing time of our method compared to our previous method, which uses individual depth images. The time is measured on a PC with an

Table 4.1: Average running time in seconds of our method and our previous method, with the input of different garment types. The time excludes the reconstruction time.

Garment	Previous Method	New Method
Sweater	46	**0.30**
Jeans	42	**0.20**
Shorts	71	**0.22**

Table 4.2: Comparison on average Geodesic Error for different types of garments. The unit is cm. Ours (No DA) stands for our method without domain adaptation.

Garment	Previous Method	New Method (No DA)	New Method
Sweater	16.05	18.64	**13.61**
Jeans	10.89	14.31	**9.70**
Shorts	22.44	25.78	**17.82**

Intel i7 3.0 GHz CPU, and shown in Table 4.1. We can see that our method demonstrates orders of magnitude speed-up against the depth-image based method, which verifies our advantages from the efficient 3D reconstruction, feature extraction and matching. The main bottleneck of our previous method is SIFT extraction and sparse dictionary learning. Our method also shows better stability in running time, especially on the shorts input, while our previous method requires more time, especially when the depth input has rich textures.

4.2.4 Generality to Novel Garments

Though we use a relatively small garment database for our experiments, we notice that our simulated models can also be generalized to recognize similar but unseen garments. For example, long-sleeve shirts and jackets can be considered as similar garments to our sweater model. Also, knit pants and suit pants are similar to our jeans model. Although they are made of different materials, the way they deform are similar to our training models in some poses. Figure 4.8 shows some extra examples of recognizing poses of unseen garments using the same weight $\mathbf{w}$ learned on our original

Table 4.3: Classification accuracy of our method on the task of garment categorization.

	Sweater	Jeans	Shorts
Accuracy	85.7%	70.0%	90.0%

dataset. We also notice that there exist some decorations such as pockets or shoulder boards on those garments, however, our method is robust enough to ignore these subtler features.

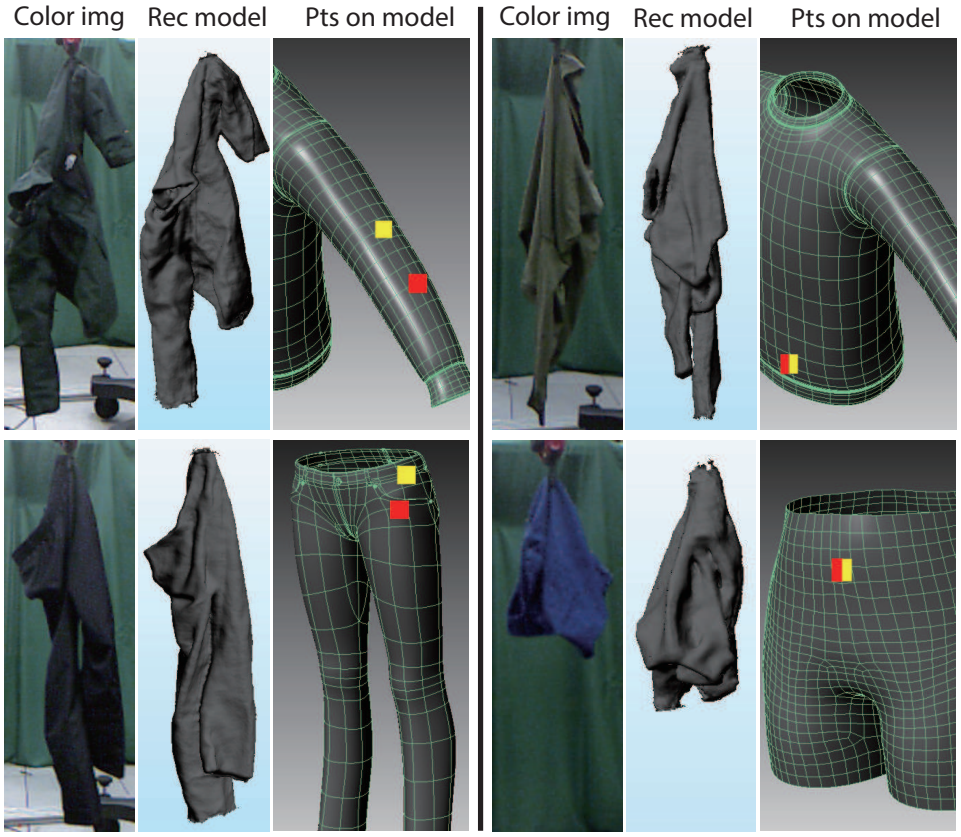


Figure 4.8: Sample results of applying our method on novel garments. Each group of results shows the color image, reconstructed model, predicted grasping points (red) vs. ground truth (yellow) marked on the model from left to right. (Best viewed in color)

4.2.5 Application to Category Classification

A basic assumption of our method is that the category of the garment is known beforehand, so that we only need to search within the training data of the same category. But our method of Nearest Neighbor (NN) search can also be used to predict the category of the garment. Therefore we also test the performance of our method on this task, by searching the NN within the entire training set instead of only the part with the same category. By adopting the NN's category as the prediction, we are able to compute the classification accuracy for evaluation, as shown in Table 4.3. We can see that our method is able to produce reasonable categorization results even without special optimization on the task.

4.3 Summary

In this chapter, we present a novel volumetric approach for the problem of pose recognition of deformable objects using a low-cost depth sensor. We first reconstruct a mesh model from the depth images, extract a volumetric 3D feature from the model, and then match it against a simulated database with a learned distance metric to find the nearest neighbor, whose grasping point will be adopted as the prediction. Experiments demonstrate superior effectiveness and efficiency of our approach.

Part II

Manipulation of Deformable Objects

Chapter 5

Regrasping and Unfolding

We are interested in unfolding the garment efficiently as a part of a pipeline for manipulating deformable garments. After several steps of regrasping, the robot holds the garment at two desired positions. Using a sweater as an example, we defined the optimal grasping positions on the two sleeves, respectively. The regrasping is built on the recognition pipeline described in our previous work [Li *et al.*, 2014b], which applies a volumetric approach to identify the garment pose using a 3D reconstructed mesh model. We extend this previous work by developing a registration-search framework that looks for an optimal position over the entire mesh model. This position is then adopted as a regrasping point in 3-dimensional space and guides the other hand to approach and regrasp. The complete pipeline of a robot folding a garment from a random state is shown in Figure 1.1, where our work described in this chapter spans initial grasping, pose estimation, regrasping for unfolding, and placing the garment flat on a table.

In this chapter, we implement a system that combines recognition and manipulation to unfold a deformable garment into a desired configuration. Key contributions are:

- A constrained weighted metric for evaluating grasping points during regrasping, which can also be used for a convergence criterion

- A fast, two-stage deformable object registration algorithm that integrates off-line simulated results with online localization and uses a novel non-rigid registration method to minimize energy differences between source and target models

- A semantically labeled garment database built with off-line simulation that contains 37 differ-

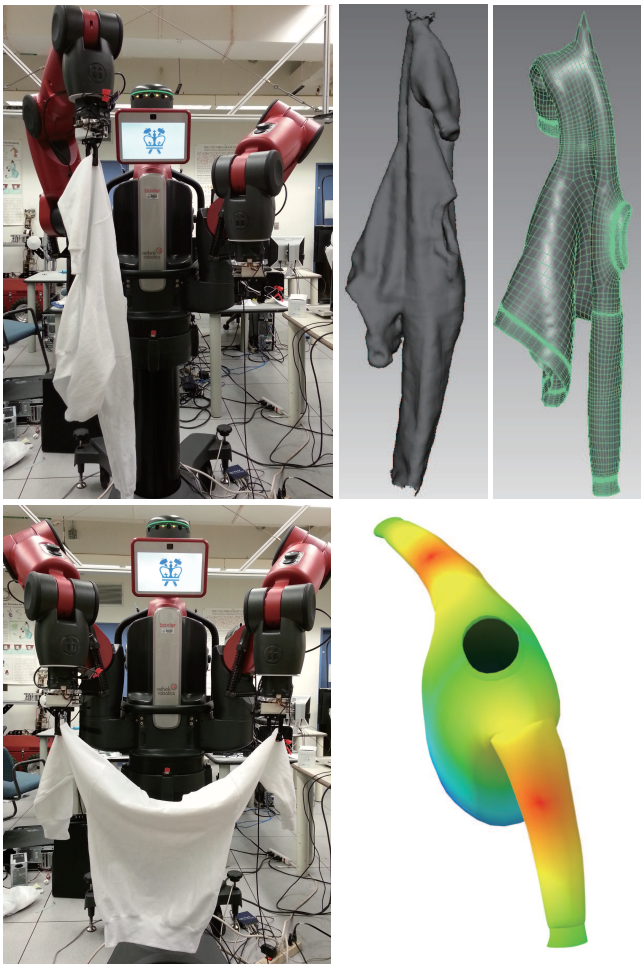


Figure 5.1: Our application scenario: a Baxter robot picks up a garment (i.e., a sweater), and a Kinect captures depth images to recognize the pose of the sweater. By deformable registration between the simulated mesh (top right) and reconstructed mesh (top middle), we obtain the regrasping point via a pre-determined point on the simulated mesh. These regrasping points are a set of pre-determined points sequence to unfold a garment. A sweater mesh with rendered weighted Gaussian distribution is shown on the bottom right. Reddish color indicates a higher score for evaluation of the grasping points, which are designated as the elbows of the sleeves. The further the grasping point from the reddish color, the lower its score, as shown in blue color. The final unfolding result by the Baxter robot is shown on the bottom left.

ent garments such as sweaters, pants, shorts, dresses, scarves, etc. The database is created by deforming garments over a large space of potential grasping points, building a unique feature

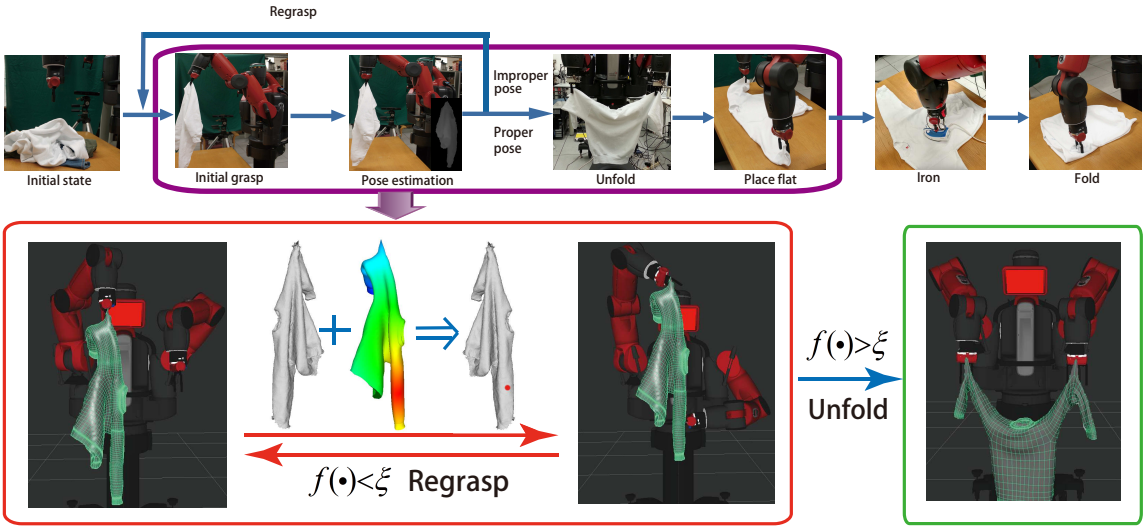


Figure 5.2: TOP ROW: The entire pipeline of dexterous manipulation of deformable objects. In this chapter, we combine the phases of initial grasp, pose estimation, regrasping, unfolding and placing the garment flat, as highlighted in the purple rectangle. BOTTOM ROW: If the recognition is not successful or the pose is improper evaluated by the $f(\cdot)$ function, the robot will regrasp the object and repeat the step of pose estimation (the red rectangle). Note that by registration between the reconstructed mesh from the Kinect and the simulate mesh from pose estimation, the robot knows where to regrasp subsequently as indicated by a red dot. This will be evaluated by the $f(\cdot)$ function. If $f(\cdot) < \xi$, the robot moves to unfold phase (the green rectangle). If this is not the case, the robot regrasps and goes back to pose estimation.

vector for each garment configuration

- Experimental results showing that our method applied on a Baxter robot is able to recognize a garment from an arbitrary pose, successfully regrasp it to put it into a known configuration, and place the garment flat on a table, which can then be folded in a subsequent process

5.1 Problem Formulation

5.1.1 Framework

Our objective is to put the garment into a certain *configuration*, which is defined as the relative grasping points on the garment [Li *et al.*, 2014b], such that the garment can be easily placed flat on a table for the folding process. This problem can be formulated as a mathematical optimization problem:

$$\max_{\mathbf{x}_L, \mathbf{x}_R} f(\mathbf{x}_L, \mathbf{x}_R). \tag{5.1}$$

Here $\mathbf{x}_L, \mathbf{x}_R \in \mathbb{R}^2$ [1] are the positions of the left and right grasping points on the garment (the configuration) and the function f is an evaluation function for such a configuration. We seek a principled way to build a feedback loop for garment regrasping, which allows us to grasp at predetermined points on the garment, and place it flat.

Suppose the candidate garment is a sweater which we want to unfold and place flat. A desired solution is grasping points $(\mathbf{x}_L^*, \mathbf{x}_R^*)$ lying on the elbows of the sleeves. Our goal is to find a pair of grasping points $(\mathbf{x}_L, \mathbf{x}_R)$, through a series of regrasping procedures that will converge to a value close to $(\mathbf{x}_L^*, \mathbf{x}_R^*)$.

5.1.2 Optimization Objective

We need a quantitative function defined on the *pose* of the garment (i.e., where the robot arm grasps the garment) in order to evaluate how good a grasping point is. While this can be computed on the continuous surface of the garment, we can also discretize the garment into a set of *anchor points* S_g, which typically contain about 15 points for a garment. After such quantization, the garment pose recognition can be treated as a discrete classification problem, which the current robotics system is able to handle reliably. This also simplifies the definition of the objective function, which then becomes a 2D score table or a matrix, given our robot has two arms.

Although the configuration variables $\mathbf{x}_L, \mathbf{x}_R$ can only be chosen from a fixed set S_g for the sake of reliable pose estimation, we use a smooth probabilistic Gaussian model to describe the objective

[1] Each garment mesh is defined in a UV 2-dimensional parameter space. When we choose a grasping point, we choose a particular set of UV parameters, which then will be mapped by registration with the sensed garment to a grasping point in $\mathbb{R}^3$.

function to make the optimization efficient. More specifically, given known locations of grasping points $\mathbf{x}_L, \mathbf{x}_R$, we treat the left and right hand of the robot independently and use the product of two Gaussians as the objective function as in Figure 5.1:

$$f(\mathbf{x}_L, \mathbf{x}_R) = \mathcal{N}\big(\mathbf{x}_L \mid \mathbf{x}_L^*, \Sigma_L\big) \cdot \mathcal{N}\big(\mathbf{x}_R \mid \mathbf{x}_R^*, \Sigma_R\big) \tag{5.2}$$

where, $\mathbf{x}_L, \mathbf{x}_R \in S_g$. Here $\mathcal{N}(\mathbf{x} \mid \mathbf{x}^*, \Sigma)$ is a 2D Gaussian distribution with $\mathbf{x}^*$ as the mean vector, and Σ as covariance of the 2D Gaussian distribution on the UV mapping. Typically, $\mathbf{x}_L^*, \mathbf{x}_R^*$ are the expected grasping points in the target configuration, and Σs control the tolerance and convergence speed (or how many times we need before reaching the target configuration) of the regrasping process.

In practice, due to the sensor errors of occlusion and deformation, it is not feasible to get a deterministic noise-free observation of $\mathbf{x}_L$ and $\mathbf{x}_R$. Instead, our previous work [Li *et al.*, 2014b] provides a probabilistic distribution $p(\mathbf{x}_l, \mathbf{x}_r \mid y)$, in which y is the current reconstructed mesh model. p is the probabilistic measurement of the confidence with which we have found the current grasping point, computed as a measure over the feature space of the current reconstructed mesh model vs. the predicted database model. Therefore according to the conditional probabilistic decomposition, the objective can be rewritten as,

$$f(\mathbf{x}_L, \mathbf{x}_R|y) = \sum_{\mathbf{x}_l, \mathbf{x}_r \in S_g} \mathcal{N}\big(\mathbf{x}_l|\mathbf{x}_L^*, \Sigma_L\big)\mathcal{N}\big(\mathbf{x}_r|\mathbf{x}_R^*, \Sigma_R\big)p(\mathbf{x}_l, \mathbf{x}_r|y). \tag{5.3}$$

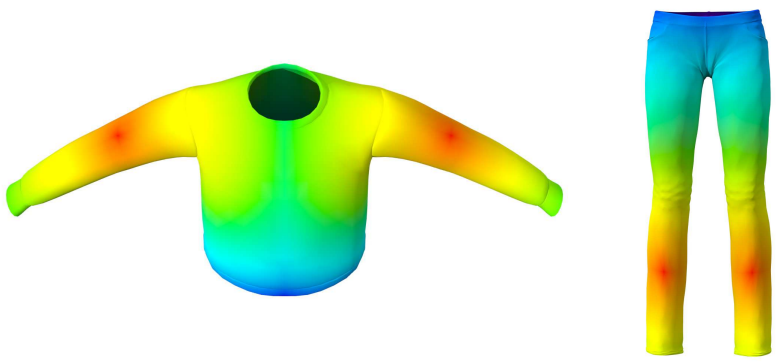
Intuitively, this objective function measures the closeness of each grasping point to the optimal grasping points, scaled by the predicted probability of the current grasping points being accurate. To simplify the form, we substitute the Gaussian Probability Density Function $\mathcal{N}(\mathbf{x}|\mathbf{x}^*, \Sigma)$ as

$$\frac{1}{2\pi\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^T\Sigma^{-1}(\mathbf{x} - \mathbf{x}^*)\right), \tag{5.4}$$

and do the optimization on $\ln f(\cdot)$. Therefore we have,

$$\begin{aligned} \ln f(\mathbf{x}_L, \mathbf{x}_R) = \sum_{\mathbf{x}_l, \mathbf{x}_r \in S_g} \Big(& -(\mathbf{x}_l - \mathbf{x}_L^*)^T\Sigma_L^{-1}(\mathbf{x}_l - \mathbf{x}_L^*) \\ & -(\mathbf{x}_r - \mathbf{x}_R^*)^T\Sigma_R^{-1}(\mathbf{x}_r - \mathbf{x}_R^*) \quad + \ln p(\mathbf{x}_l, \mathbf{x}_r|y)\Big). \end{aligned} \tag{5.5}$$

Experimentally, we use the geodesic distance on the garment surface to measure the quality of the regrasping. In our settings, there is no reason to have more weights in one direction against



(a) (b)

Figure 5.3: Visualization of the defined objective used in this chapter. (a) A sweater model rendered with weighted Gaussian distribution. (b) A pants model rendered with weighted Gaussian distribution. When a point is given over the garment surface, we can then evaluate the score by the objective function $f(\cdot)$.

another. We compute the covariance matrix as:

$$\Sigma_L^{-1} = \begin{bmatrix} \sigma_l & 0 \\ 0 & \sigma_l \end{bmatrix}, \quad \Sigma_R^{-1} = \begin{bmatrix} \sigma_r & 0 \\ 0 & \sigma_r \end{bmatrix}, \tag{5.6}$$

Subsequently, the objective function which needs to be maximized becomes:

$$\ln f(\mathbf{x}_L, \mathbf{x}_R) = - \sum_{\mathbf{x}_l, \mathbf{x}_r \in S_g} \Big(\sigma_l \|\mathbf{x}_l - \mathbf{x}_L^*\|^2 \quad + \sigma_r \|\mathbf{x}_r - \mathbf{x}_R^*\|^2 - \ln p(\mathbf{x}_l, \mathbf{x}_r | y) \Big). \tag{5.7}$$

The related parameters in the objective such as σ_L and σ_R are set depending on the desired configuration. For example, for sweaters, we set $\mathbf{x}_L^*$ and $\mathbf{x}_R^*$ on the elbow of the two sleeves. The Gaussian formulation ensures a smooth decrease from the expected grasping points, as visualized in Figure 5.1 as an example.

5.2 Implemented Method

5.2.1 Deformable Registration

After obtaining the location of the current grasp point, we seek to register the reconstructed 3D model to the ground truth garment mesh to establish point correspondences. In terms of the material

property of the garment, the closest matched mesh model from the database sometimes is not very similar to the reconstructed mesh. To find the best correspondence, the registration is employed here. The input to the registration is a canonical reference ("source") triangle mesh S that has been computed in advance and stored in the garment database, and a target triangle mesh T representing the geometry of the garment grasped by the robot, as acquired by 3D scans of the grasped garment.

The registration proceeds in three steps. First, we scale the source mesh S to match its size to the target mesh T. Next, we apply an iterative closest point (ICP) technique to rigidly transform the source mesh S (i.e., via only translation and rotation). Finally, we apply a non-rigid registration technique to locally deform the source mesh S toward the target T.

Scaling First, we compute a representative size for each of the source and target meshes. For a given mesh, let a_i and $\mathbf{g}_i$ be the area and barycenter of the ith triangle. Then the area-weighted center $\mathbf{c}$ of the mesh is

$$\mathbf{c} = \sum_i^{N_S} a_i \mathbf{g}_i \Big/ \sum_i^{N_S} a_i, \tag{5.8}$$

where N_S is the number of vertices of the source mesh S. Given the area-weighted center, the representative size l of the mesh is given by

$$l = \sum_i^{N_S} a_i \|\mathbf{g}_i - \mathbf{c}\| \Big/ \sum_i^{N_S} a_i. \tag{5.9}$$

Let the representative sizes of the source and target meshes be l_S and l_T, respectively. Then, we scale the source mesh by a factor of l_T/l_S.

Computing the rigid transformation We use a variant of ICP [Besl and McKay, 1992] to compute the rigid transformation. ICP iteratively updates a rigid transformation by (a) finding the closest point $\mathbf{w}_j$ on the target mesh T for each of the vertices $\mathbf{v}_j$ of the source mesh S, (b) computing the optimal rigid motion (rotation and translation) that minimizes the distance between $\mathbf{w}_j$ and $\mathbf{v}_j$, and then (c) updating the vertices $\mathbf{v}_j$ via this rigid motion.

To accelerate the closest point query, we prepare a grid data structure during preprocessing. For each grid point, we compute the closest point on the target mesh using fast sweeping [R.Tsai, 2002], and store for runtime using both the found point and its distance to the grid point as shown in Figure 5.4. At runtime, we approximate the closest point query for vertex $\mathbf{v}_j$ by searching only among those eight precomputed closest points corresponding to the eight grid points surrounding $\mathbf{v}_j$, thereby reducing the complexity of the closest point query to $O(1)$ per vertex.

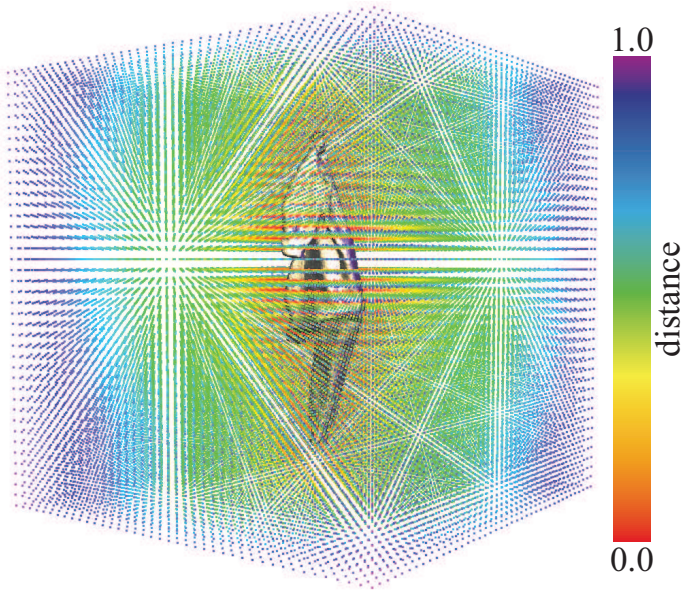


Figure 5.4: Visualization of distance function given a mesh. The color bar on the right shows the normalization distance.

After establishing point correspondences, we compute the optimal rotation and translation for registering $\mathbf{v}_j$ with $\mathbf{w}_j$ [Besl and McKay, 1992]. We iteratively compute point correspondences and rigid motions until successive iterations converge to a fixed rigid motion, yielding a rigidly registrated source mesh $\bar{S}$.

Non-rigid registration Given a candidate source mesh $\bar{S}$ obtained via rigid registration, our non-rigid registration seeks the vertex positions $\mathbf{v}_j$ of the source mesh S that minimize

$$E_{\bar{S},T}(S) = E_{\text{fit}}(S,T) + E_{\text{def}}(S,\bar{S}), \tag{5.10}$$

where $E_{\text{fit}}(S,T)$ penalizes discrepancies between the source and target meshes, and $E_{\text{def}}(S,\bar{S})$ seeks to limit and regularize the deformation of the source mesh away from its rigidly registrated counterpart $\bar{S}$. The term

$$E_{\text{fit}} = \sum_{i=1}^{N_S} (\text{dist}(\mathbf{g}_i))^2 \, \bar{A}_i, \tag{5.11}$$

penalizes deviation of the source and target meshes. Here $\mathbf{g}_i$ is the barycenter of the triangle i, and $\text{dist}(\mathbf{g}_i)$ is the distance from $\mathbf{g}_i$ to the closest point on the target mesh. As in the rigid case, we use the precomputed distance field to query for the distance.

It might appear that the fitting energy E_{fit} could be trivially minimized by moving each vertex of mesh S to lie on mesh T. In practice, however, this does not work because all of the geometry of the precomputed reference mesh $\bar{S}$ is discarded; instead, the geometry of this mesh, which was precomputed using fabric simulation, should serve as a prior. Thus, we introduce a second term to retain as much as possible the geometry of the reference mesh $\bar{S}$:

The deformation term $E_{\text{def}}(S, \bar{S})$, derived from a physically based energy (e.g., see [Grinspun *et al.*, 2003]), is a sum of three terms

$$E_{\text{def}}(S, \bar{S}) = \kappa E_{area} + \beta E_{angle} + \alpha E_{hinge}, \tag{5.12}$$

where α, β and κ are user-specified coefficients.The term

$$E_{area} = \sum_{i=1}^{N_S} \frac{1}{2} \left(\frac{A_i}{\bar{A}_i} - 1 \right)^2 \bar{A}_i, \tag{5.13}$$

penalizes changes to the area of each mesh triangle. Here A_i is the area of the triangle i, and $\bar{\cdot}$ refers to a corresponding quantity form the undeformed mesh $\bar{S}$. The term

$$E_{angle} = \sum_{i=1}^{N_S} \sum_{k=1}^{3} \frac{1}{6} \left(\frac{\theta_{ik}}{\bar{\theta}_{ik}} - 1 \right)^2 \bar{A}_i, \tag{5.14}$$

penalizes shearing of each mesh triangle, where θ_{ik} is the kth angle of the triangle i. The term E_{hinge} [Grinspun *et al.*, 2003]

$$E_{hinge} = \sum_{e} (\theta_e - \bar{\theta}_e)^2 \|\bar{e}\| / \bar{h}_e, \tag{5.15}$$

penalizes bending, measured by the angle formed by adjacent triangles. Here θ_e is the *hinge angle* of edge e, i.e., the angle formed by the normals of the two triangles incident to e; $\|\bar{e}\|$ is the length of the edge e, and $\bar{h}_e$ is a third of the sum of the heights of the two triangles incident to the edge e.

We used the secant-version of the L-M method[Madsen *et al.*, 2004] to seek the source mesh S that minimizes the energy Eq.(7.9). Sample registration results are shown in Figure 5.5.

5.2.2 Grasping Point Localization

We use a pre-determined anchor point (e.g., elbow on the sleeve of a sweater) to indicate a possible regrasping point. The detection of the regrasping point can be summarized in two steps: *global localization* and *local refinement*.

Global localization is achieved by deformable registration. The registered simulation mesh will provide a 3D regrasping point from the recognized state which will be then mapped onto the reconstructed mesh.

In order to improve the regrasping success rate, we propose a step of *local refinement*. The point on the actual garment may be hard to grasp for several reasons. One is that during the garment manipulation steps, such as rotation, the curvature over the garment may change. Another reason is that when considering the width of robot hand gripper, a ridge curve with proper orientation and width should be selected for regrasping. We consider the proper orientation as a direction perpendicular to the opening of the gripper. Therefore, we propose an efficient 1D blob curvature detection algorithm that can find a refined position in the local area over the garment surface via an IR range sensor.

In our experiment, the Baxter robot is equipped with a IR range sensor close to the gripper as shown in Figure 5.6 top. Once the gripper moves to the same height of the predicted 3D regrasping point from registration, it will perform a horizontal scan search, moving from one side to the other, so that the IR sensor will scan over the full local curvature.

We then apply a curvature detection algorithm that convolves the IR depth signal with a fixed width kernel, where the width is determined by the opening of the gripper. Here we use a Laplacian of Gaussian Kernel $g''(x)$:

$$g''(x) = (\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2})e^{-\frac{x^2}{2\sigma^2}} \tag{5.16}$$

where x is the depth signal, and σ is the width parameter.

5.2.3 Convergence

After the regrasping is finished, we evaluate the current grasping configuration by the objective function $f(\cdot)$. If $f(\cdot)$ is greater than a given value ξ, which means the grasping points are on the desired positions, and the robot will then stop regrasping and enter the placing flat mode. The two arms will open to slightly stretch the garment and place it on a table. The overall algorithm is summarized in Algorithm 3.

Algorithm 3: Iterative Procedure for Regrasping

Input: Simulation meshes = $M = \{M^1_{sim}, M^2_{sim}, ..., M^n_{sim}\}$;
Trained classifier C;
Objective function f;

Output: Two desired grasping points X_L and X_R;

$f_{score} \leftarrow \infty, i \leftarrow 1$;
while $f_{score} < \xi$ **do**
  Pick up at a grasping point P^i_{grasp};
  $M^i_{rec} \leftarrow$ 3D Reconstruction;
  $M^i_{sim} \leftarrow C(M^i_{rec}; M^1_{sim}, M^2_{sim}, ..., M^n_{sim})$ //Recognition;
  $P^{i+1}_{grasp} \leftarrow$ Reg(M^i_{rec}, M^i_{sim}) //Registration;
  $f_{score} \leftarrow f(P^i_{grasp}, P^{i+1}_{grasp})$;
  $i \leftarrow i + 1$;
return $\mathbf{x}_L$ *and* $\mathbf{x}_R$;
Place the garment flat on a table;

5.3 Experimental Results

To evaluate our results, we tested our method on several different garments such as sweater and pants for multiple trials, as shown in Figure 7.10 left. A high resolution video of our experimental results is online at http://www.cs.columbia.edu/~yli/laundry_robot.

5.3.1 Robot Setup

In our experiments, we use a Baxter research robot, which is equipped with two arms with seven degrees of freedom. To improve grasp stability and form a closed loop controller, we add tactile sensors to each of the grippers as shown in Figure 5.6. Each robot hand is equipped with a IR range sensor (come with the Baxter robot), which is used for local curvature detection, as described in section 5.2.2. A Kinect sensor is mounted on a fixed frame at a height of 1.2 meters to capture the depth images.

5.3.2 Garment Database and Real-time Recognition

Our recognition method consists of two stages, offline model simulation and online recognition. In the offline model simulation stage, we use a physics engine [Maya, 2014] to simulate the stationary state of the mesh models of different types of garments in different poses. More specifically, we manually label each garment in the database with the key grasping points such as sleeve end, elbow, shoulder, chest, and waist, etc. For each grasping point, we compute the garment layout by hanging under gravity in the simulator. The simulated meshes are used as the training data for 3D shape-based matching and pose recognition [Li *et al.*, 2014b]. In our experiments, we assume the category of the garment is known. Therefore, we start with the garment pose estimation.

Below we summarize the pose recognition method, details can be found in [Li *et al.*, 2014b]. We first pick up the garment at a random point. In the online recognition stage, we use a Kinect sensor to capture depth images of different views of the garment while it is being rotated by a robotic arm. The garment is rotated 360° clockwise and then 360° counter-clockwise to obtain about 550 depth images for an accurate reconstruction. We reconstruct a 3D mesh model from the depth image segmentation and volumetric fusion. Then with an efficient 3D feature extraction algorithm, we build up a binary feature vector and finally match against the offline database for pose recognition. One of the outputs is a high-quality reconstructed mesh, which is used for 3D registration and accurate regrasping point prediction, as described below.

5.3.3 Registration

As described in section 5.1, we apply both rigid and non-rigid registrations. The rigid registration step mainly focuses on mesh rescaling and alignment, whereas the non-rigid registration step refines the results and improves the mapping accuracy. In Figure 5.5, we compare the difference between using rigid registration only and using rigid plus non-rigid registration side by side. We can clearly see that with non-rigid registration, the two meshes are registered more accurately. In addition, the location of the designated grasping points on the sleeves are also closer to the ground truth points. Note that for the fourth row, after the alignment by the rigid registration algorithm, the state is evaluated as a local minimum. Therefore, there is no improvement by the following non-rigid registration. But as we can see from the visualization, such a case is still good enough for finding point correspondence.

Figure 5.5: Registration examples. FIRST ROW: A sweater grasped at elbow. SECOND ROW: A sweater grasped at sleeve end. THIRD ROW:A pair of pants grasped near knee. FOURTH ROW: A pair of pants grasped near ankle. Each row depicts from left to right: a reconstructed mesh, the predicted mesh from the database, rigid registration only, and rigid plus non-rigid registration.

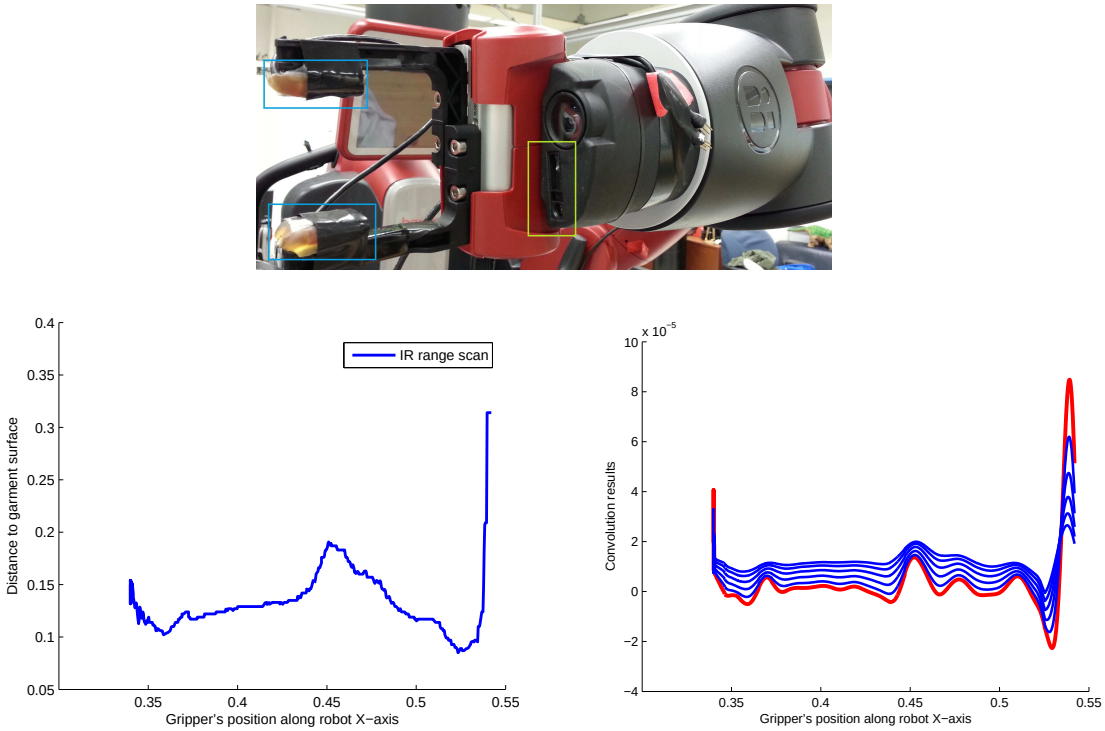


Figure 5.6: IR range sensor scan example. TOP: An image of the Baxter hand. The IR range sensor is shown in yellow rectangle and two Tactile sensors in blue rectangles. BOTTOM LEFT: Single reading plot from the IR range sensor. BOTTOM RIGHT: Convoluted result of the sensor reading and a Laplacian-Gaussian kernel with different kernel size, as indicated in blue and red curves. The lowest point (in red) is the place the gripper should grasp.

5.3.4 Search for Best Grasping Point by Local Curvature

Once we choose a potential grasping point, we can perform a search to find the best local grasping point for the gripper. We are trying to a find a fold in the vicinity of the potential grasping point with a high local curvature tuned to the gripper width that allows for a stable grasp. The opening size of the gripper is approximately $8cm$ and empirically we set $\sigma = 10$ in the Equation 5.16. Figure 5.6 top shows a picture of the IR range sensor on the gripper. A plot of its signal, as well as the convoluted signal, are shown in Figure 5.6 bottom left and right. The blue and red curves correspond to the different kernel sizes in Equation 5.16. The red color curve indicates the best fitting parameter ($\sigma = 10$) according to the opening size of the grippers. We can clearly see that the response from the filter is at a minimum where the grasping should take place. The tactile sensors then assure that

the gripper has properly closed on the fabric.

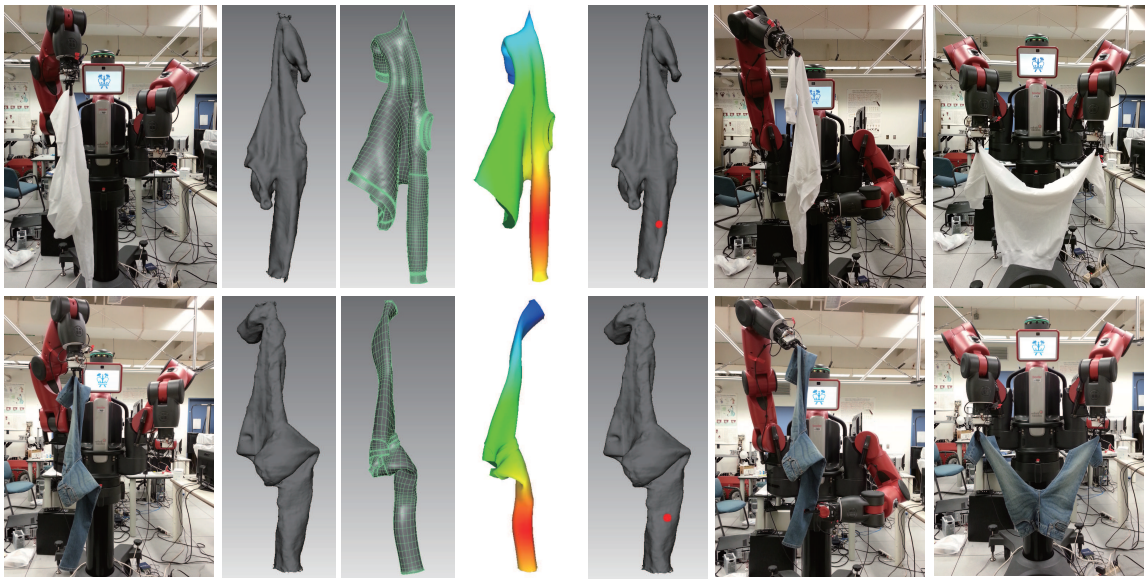


Figure 5.7: Examples of each step in our unfolding procedure. For each row from left to right is: a snapshot of initial pick up, a 3D reconstructed mesh, a predicted mesh from database, the predicted mesh with weighted Gaussian distribution distance, predicted regrasping point on the 3D reconstructed mesh, a snapshot of regrasping, and finally a snapshot of unfolding. TOP ROW: The Baxter robot unfolds a sweater following pick up. BOTTOM ROW: The Bater robot unfolds a pair of pants following pick up.

5.3.5 Iterative regrasping

Figure 5.7 shows two examples (sweater and pants) of iterative regrasping using the Baxter robot. The robot first picks up a garment at a random grasping point. Once the arm reaches a pre-defined position, the last joint of the arm starts to rotate and the Kinect will capture the depth images as it rotates, and reconstruct the 3D mesh in real-time. After the rotation, a predicted pose is recognized [Li *et al.*, 2014b] as shown in the third image of each row. For each pose, we have a constrained weighted evaluation metric over the surface to identify the regrasping point as indicated in the fourth image. By registration of the reconstructed mesh and predicted mesh from the database, we can map the desired regrasping point onto the reconstructed mesh. The robot then regrasps by moving the other gripper towards it. With our 1D blob curvature detection method, the

Garment	# of Trial	Successful Recognition	Successful Regrasping	Successful Unfolding	Avg. # of Regrasps Success Only
Sweatshirt	10	9/10	8/10	8/10	1.6
Sweater	10	8/10	8/10	7/10	1.6
Knitwear	10	9/10	9/10	8/10	1.7
Jeans	10	9/10	9/10	9/10	1.3
Pants	10	8/10	10/10	9/10	1.4
Leggings	10	8/10	9/10	8/10	1.4
Shorts	10	7/10	8/10	7/10	1.9
Average	10	8.3/10	8.7/10	8.0/10	1.6

Table 5.1: Results for each unfolding test on the garments. We evaluate the results by recognition, regrasping, unfolding, and regrasping attempts for each test. The last row shows the average of each evaluation component.

gripper can move to the best curvature on the garment and regrasp, which increases the success rate. The iterative regrasping stops when the two grasped points are the designated anchor points on the garment (e.g., elbows on the sleeves of a sweater).

Figure 5.8: A picture of our test garments.

Figure 5.8 left shows 7 sample garments in our test, and the Table 5.3.5 shows the results. For each garment, we perform 10 unfolding tests. We have on average an 83% successful recognition rate for the pose of the objects over all the garments. We have on average an 87% successful regrasping rate for each garments, where regrasping is defined as a successful grasp of the other

arm on the garment. 80% of the time we are able to successfully unfold the garment, placing the grippers at the designated grasping points.

Unsuccessful unfolding occurred when either the gripper lost contact with the garment, or the gripper was unable to find a regrasping point. Although we did not perform this experiment, it is possible to restart the method after one of the grippers loses contact as an error recovery procedure.

For the successful unfolding cases, we also report the average number of regrasping attempts. The minimum number of regrasping attempts $= 1$. This happens when the initial grasping is at one of the desired positions, and the regrasping succeeds at the other desired position (i.e., two elbows on the sleeves for a sweater). In most cases, we are able to successfully unfold the garments using $1 - 2$ regraspings.

Among all these garments, jeans, pants, and leggings achieve high success rate because of their unique layout when grasping at the leg position. The shorts are difficult for both recognition and unfolding steps possibly because its ambiguous appearances in different grasping points. One observation is that in a few cases, when the recognition is not accurate, our registration algorithm was sometimes able to find a desired regrasping point for unfolding. This is an artifact of the geometry of pant-like garments where the designated regrasping points are at the extreme locations on the garments.

We also show that after grasping at two desired points, the robot will proceed to place the garment on a table. In our experiments, we use cardboard to simulate a table area. As shown in Figure 5.9, the robot is able to place the garment flat with a simple move when grasping at a pair of two desired grasping points. With such a flat configuration, the robot can begin to fold it, which is described in Chapter 7.

5.4 Summary

In this chapter, we propose a novel solution for the problem of unfolding a garment to a desired configuration via regrasping. The unfolding procedure contains garment pick up, pose estimation, regrasping, and placing flat on a table. We use simulated predictive thin shell models for garments to automatically create a large database of garments and their poses, which can be used in a learning algorithm to find object pose. We also create a real-time 3D volumetric model from 3D scanning

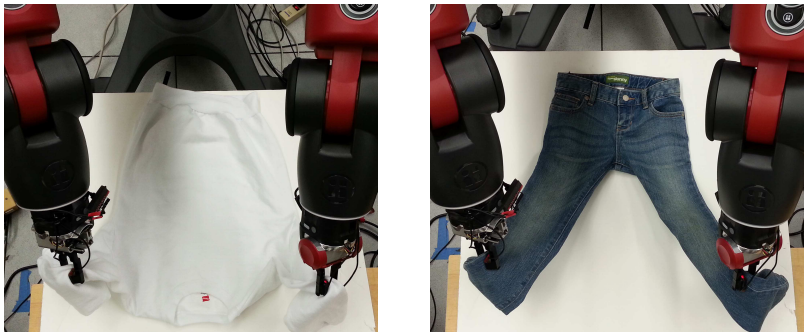


Figure 5.9: The Baxter robot places a garment flat on a table. LEFT: The garment is a sweater and the two desired grasping points are on the sleeves. RIGHT: The garment is a pair of pants and the two desired grasping points are on the lower leg parts.

which can be used for registration with the simulated models. The regrasping point is found by a fast, two-stage deformable object registration algorithm that uses a novel non-rigid registration method to minimize energy differences between source and target mesh models The unfolding state is determined by a constrained weighted objective function for the current two grasping points. Experimental results show that our method is able to unfold a number of common garments with a high success rate.

Chapter 6

Ironing

During the manipulation of deformable garments, wrinkles may be created, which will affect further actions in the pipeline, such as folding. Therefore, we consider using robotic ironing to handle these scenarios. Figure 6.1 shows some examples of robotic ironing. The top row shows a case of ironing onto a height bump. If ironing a height bump, we can remove the air trapped in the bump, but permanent wrinkles could also be created. This implies that we need to avoid ironing a height bump. The bottom row shows a comparison of pulling the boundaries of the fabric, and ironing for wrinkle removal. We can see that pulling can only remove some height bumps (smooth regions) but not permanent wrinkles. But with further ironing, all the permanent wrinkles can be removed. Figure **??** shows a entire workflow of our robotic ironing. Within the right red box, we first detect height bumps using the depth fusion algorithm. Then applying GMM to fit those height bumps. We also employ diffuse reflection to detect surface discontinuities. Combining these two results, permanent wrinkles are detected using our probabilistic multi-sensor framework. Finally, we export and adapte the detected wrinkles to the robot world in the left green box. All the detected wrinkles are sorted in an optimized way and exported to the Baxter robot for ironing.

Permanent wrinkles correspond to high curvatures or discontinuities in the gradient of the garment shape. Other deformed regions are much smoother, and their height deviations are larger. An off-the-shelf sensor, like Kinect, is useful for capturing regions with large variation in height. But it is usually too low resolution for capturing smaller surface discontinuities. On the other hand, because of the rapid change in the gradient (or normal) around the permanent wrinkles, there is a rapid change in the lighting effect. Therefore, an illumination based approach suffices to capture

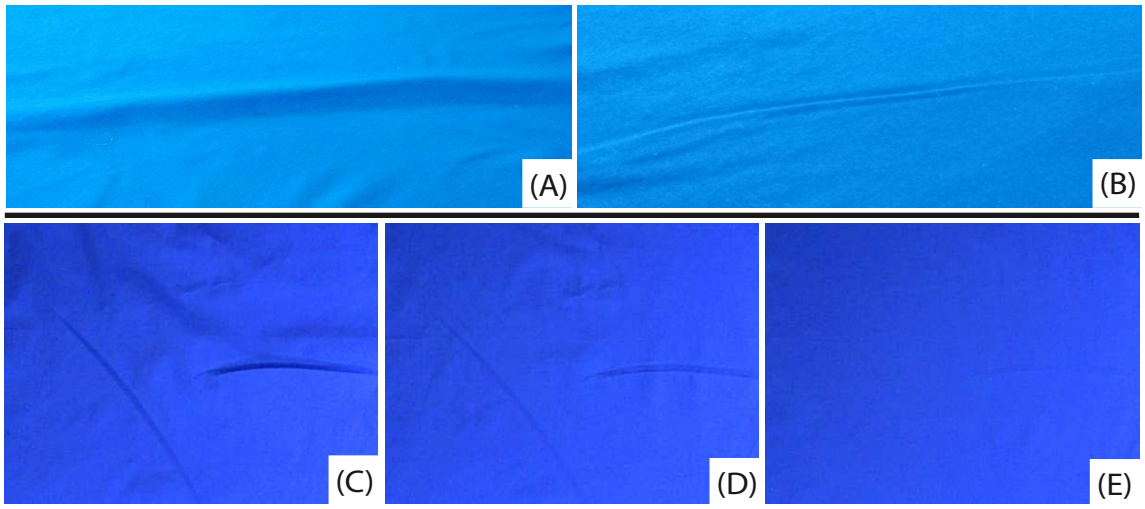


Figure 6.1: Best viewed by zooming. TOP: Effects of ironing on a bump. (A): A typical height bump on a piece of cloth. (B): Results of ironing on the height bump. BOTTOM: A comparison of effects of pulling the boundaries and ironing on wrinkles. (C): A region with height bumps and permanent wrinkles. (D): After pulling the boundaries of the region, there are still some permanent wrinkles left. (E): The results of our ironing method showing the removal of the permanent wrinkles.

the permanent wrinkles. We found that using a combination of a 3D depth scan and an illumination based approach allows us to robustly identify permanent wrinkles from other deformations without the need of a carefully calibrated system.

6.1 Problem Formulation

In our approach, we employ two different types of visual scans to detect non-smooth and smooth regions. One is a curvature scan by a Kinect sensor which fuses multiple depth images to find *height bumps* (see Sec. 6.2.1). Height bumps are non-flat regions with smooth curvature. The other one is a discontinuity scan using diffuse reflection to find *permanent wrinkles* (see Sec. 6.2.2). In terms of the scan resolution, the Kinect has relatively lower resolution in the surface reconstruction, whereas the diffuse reflection is able to capture smaller surface discontinuities. Empirically, the non-smooth regions are mostly the permanent wrinkles as we defined in Chapter 1. The height deviation of smooth regions detected by the curvature scan are likely height bumps, caused by frictional force between the garment and the table, which can be removed by iterative pulling [Sun *et al.*, 2015].

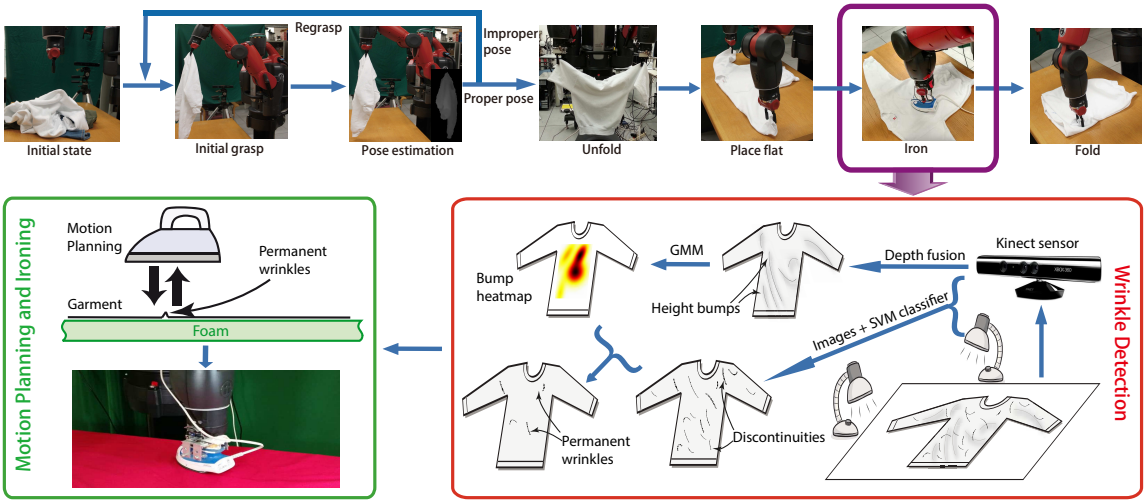


Figure 6.2: TOP ROW: The entire pipeline of dexterous manipulation of deformable objects. In this chapter, we are focusing on the phases of garment ironing, as highlighted in the purple rectangle. BOTTOM ROW: Detailed ironing process from wrinkle detection to motion planing and ironing. We first detect height bumps using the depth fusion algorithm. Then applying GMM to fit those height bumps. We also employ diffuse reflection to detect surface discontinuities. Combining these two results, permanent wrinkles are detected using our probabilistic multi-sensor framework. Finally, all the detected wrinkles are sorted in an optimized way and exported to the Baxter robot for ironing.

With the two scan methods mentioned above, we build a probabilistic multi-sensor framework that can efficiently detect and classify the permanent wrinkles that need to be ironed. Since it is expensive to set up a well-calibrated diffuse reflection system to accurately reconstruct and measure the volume of the wrinkles, we argue that our framework is more generalizable and easy to deploy.

For the discontinuity scan, we are only interested in the position of the permanent wrinkles but not full 3D reconstruction. The regions to be evaluated here are permanent wrinkles (non-smooth regions), which are from the discontinuity scan. Let $\mathcal{K}$ represent a set of the detected height bumps from the curvature scan and $\mathcal{D}$ represent a set of the illumination discontinuities from discontinuity scan. We first define a pixel-wise probability. The probability of an illumination discontinuity

$d_i \in \mathcal{D}$ in a permanent wrinkle set $\mathcal{W}$ is evaluated as,

$$\begin{aligned} P(d_i \in \mathcal{W}) &= \sum_{d_i} P(x \in \mathcal{W}) \\ &= \sum_{d_i} P(x \in \mathcal{D}) P(x \in \overline{\mathcal{K}}) R(d_i) \end{aligned} \tag{6.1}$$

where x represents a pixel in the image, and $R(d_i)$ evaluates the confidence of a discontinuity d_i to be classified as a permanent wrinkle in terms of the local shape features, etc. Then we further define,

$$Q(d_i \cap \overline{\mathcal{K}}) = \sum_{d_i} P(x \in \mathcal{D}) P(x \in \overline{\mathcal{K}}) \tag{6.2}$$

where $Q(d_i \cap \overline{\mathcal{K}})$ evaluates the probability of a discontinuity d_i in $\mathcal{D}$ not intersecting with height bumps using the results $\mathcal{K}$ from the curvature scan, and $\overline{\mathcal{K}}$ represents regions other than detected height bumps. From Eq. (6.2), we can infer that $Q(d_i) = 1$. And Eq. 6.1 can be written as,

$$P(d_i \in \mathcal{W}) = Q(d_i \cap \overline{\mathcal{K}}) \cdot R(d_i) \tag{6.3}$$

6.1.1 Joint probability evaluation

We could calculate $P(d_i \in \mathcal{W})$ using Eqs. (6.2) and (6.3). For robust ironing, we want to avoid ironing regions that are close to a height bump or in between adjacent height bumps. We found that using a Gaussian Mixture Model (GMM) as follows works well to approximate $P(d_i \in W)$ for our purpose. We start by first using the results from the curvature scan and then the discontinuity scan in the detection pipeline. The probability of a discontinuity d_i to be a permanent wrinkle given detected height bumps from curvature scan in set $\mathcal{K}$ is evaluated by $Q(d_i \cap \overline{\mathcal{K}})$. Using Bayes' rule, it can be written as,

$$Q(d_i \cap \overline{\mathcal{K}}) = Q(d_i | \overline{\mathcal{K}}) \cdot Q(\overline{\mathcal{K}}) \tag{6.4}$$

$Q(\overline{\mathcal{K}})$ is a joint distribution of $k_1, k_2 ... k_N \in \mathcal{K}$. We further simplify the scenario that all the height bumps are created independently. Applying the chain rule with conditional dependency assumption,

$$\begin{aligned} Q(d_i \cap \overline{\mathcal{K}}) &= Q(d_i | \overline{k}_1, \overline{k}_2 \cdots \overline{k}_N) Q(\overline{k}_1, \overline{k}_2 \cdots \overline{k}_N) \\ &= \prod_{j=1 \cdots N} Q(d_i | \overline{k}_j) = \prod_{j=1 \cdots N} (Q(d_i) - Q(d_i | k_j)) \\ &= \prod_{j=1 \cdots N} (1 - Q(d_i | k_j)) \end{aligned} \tag{6.5}$$

Considering the height bumps detected from the curvature scan are mostly smooth regions, we can formulate those regions as an hypothesis for the permanent wrinkles. The detected height bumps can be represented as several Gaussian distributions. Eq. (6.5) can be written as,

$$\prod_{j=1\cdots N}(1-Q(d_i|k_j)) = \prod_{j=1\cdots N}(1-Q(d_i|\mathcal{N}(k_j^{(x,y)},\Sigma_j^{(x,y)})) \tag{6.6}$$

where $k_j^{(x,y)}$ is the center of the height bump k_j. The covariance matrix $\Sigma_j^{(x,y)}$ is defined as,

$$\Sigma_j^{(x,y)} = \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix} \tag{6.7}$$

where D_1 and D_2 is the fitted first and the second principal axes of the segmented height bump. The closer the d_i to the height bump, the higher probability $Q(d_i|k_j)$ gets. For each detected discontinuity d_i, we will evaluate its probability to be ironed as in Eq. (6.5). This formulation can be further approximated by a Gaussian Mixture Model (GMM).

6.1.2 Individual probability evaluation

In this part, we show the inference of $R(d_i)$. The images captured from the discontinuity scan themselves carry useful information for permanent wrinkle classification, such as texture and edges. Therefore, we propose a SIFT-SVM based classifier to find permanent wrinkles at pixel level.

The training data is collected by capturing dozens of images with normalization under a Lambertian reflection model (see Sec. 6.2.2 for more details). Then we manually label those wrinkles which are considered as the permanent wrinkles in the image at pixel level to form a positive training set $\mathcal{T}_p$. We also create a negative training set $\mathcal{T}_n$ by scattering a certain amount of pixel points over the whole image. Since the wrinkles only occupy a small set of pixels in the image, all the scattered points are mostly negative samples.

For each pixel in training set $\mathcal{T}_p$ or $\mathcal{T}_n$, we compute the SIFT descriptor [Vedaldi and Fulkerson, 2008]. Then we create two training sets $\{\mathbf{X}\}^{|\mathcal{T}_\mathbf{p}|}$ and $\{\mathbf{X}\}^{|\mathcal{T}_\mathbf{n}|}$, where $\{\mathbf{X}\}^{|N|} = \{\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_N\}$ contains N sets of 128 dimensional feature vector. Each pixel will be considered as a training sample in training a SVM classifier. The output is a classification confidence score that will be assigned to each pixel $I_{u,v}$ as,

$$S(I_{u,v}) = sigmoid(\sum_{j=1}^{N} a_j y_j \kappa(x_j, x) + b) \tag{6.8}$$

where $(a_1, a_2, ..., a_N, b)$ are parameters from SVM training, and $\kappa(x_j, x)$ is a kernel function. Then the individual probability of a discontinuity to be a permanent wrinkle is calculated as a normalized summation of pixels that overlaid on itself as,

$$R(d_i) = \frac{1}{|d_i|} \sum_{I_{u,v} \in d_i} S(I_{u,v}) \tag{6.9}$$

where $|d_i|$ represents number of pixels associated with this discontinuity.

6.2 Multi-sensor Detection

From Sec. 6.1, we assume that the garment or a piece of cloth has been roughly flattened on a table, with some non-smooth and smooth regions existing. In this section, we focus on the detection of height bumps and discontinuities over the surfaces.

6.2.1 Curvature scan

Surface Reconstruction To obtain a distribution of the height bumps, a Kinect fusion algorithm [Newcombe *et al.*, 2011] is employed. The Kinect sensor projects infrared light onto the surface of the cloth and receives the reflection by a monochrome CMOS sensor, which is able to capture 3D information of the projected surface. Considering the resolution of the Kinect depth fusion stream (640×480), such fusion process will lose details of some small local curvature. However, it can robustly capture the fused height deviation over the surface, which is an important hypothesis for the probabilistic multi-senosr wrinkle detection framework, as described in Sec. 6.1.

Since the reconstruction requires disparity between each camera frame, simply rotating the Kinect along one axis is not enough. This will generate little or no parallax between depth frames, which leads to a singularity for a reconstructed 3D point. We designed a two-axis motor structure to overcome such cases, see Figure 6.3. Each time the two joints will rotate simultaneously and create enough parallax. Figure 6.3 right shows a reconstruction result from the setting.

Height bump detection As seen from Figure 6.3, right, the reconstructed mesh has some ridges and rugged terrains. To find smooth bump-like regions, the *Hessian matrix* is calculated over the

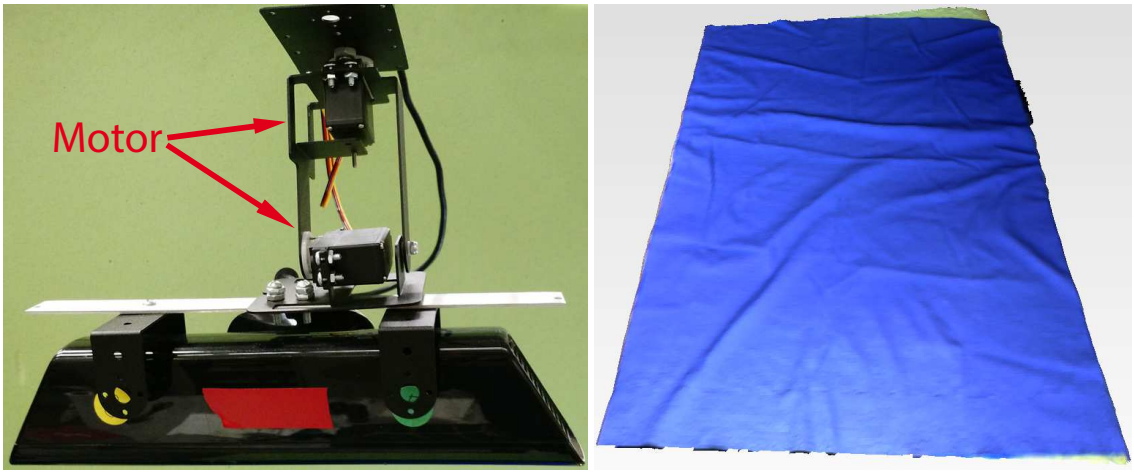


Figure 6.3: LEFT: A Kinect attached to two-axis motors rotation module. RIGHT: A reconstructed cloth surface rendered with color. Most of the height bumps are caught, but not the detailed discontinuity information.

whole image I pixel by pixel. The second order Hessian matrix $\mathbf{H}$ is defined as,

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 I}{\partial x^2} & \frac{\partial^2 I}{\partial x \partial y} \\ \frac{\partial^2 I}{\partial x \partial y} & \frac{\partial^2 I}{\partial y^2} \end{bmatrix} \tag{6.10}$$

For each pixel, we can calculate two eigenvalues λ_1 and λ_2 of $\mathbf{H}$. Assuming $\lambda_1 > \lambda_2$, then pixels that satisfied the following equation will be classified as a bump point and added to a set $\mathcal{R}$ [Koenderink and van Doorn, 1992].

$$\frac{2}{\pi}\tan^{-1}\left(\frac{\lambda_1 + \lambda_2}{\lambda_1 - \lambda_2}\right) \in [-\frac{1}{8}, \frac{5}{8}) \tag{6.11}$$

Those connected bump points in $\mathcal{R}$ can be viewed as a height bump. Since each pixel is associated with a depth value, the total volume of each region can be estimated by summation of all normalized depth values within the region. Furthermore, all the bumps can be ranked in terms of their volume, and large volume smooth regions will be defined as the height bumps. Given a cluster of all ridge points associated with a height bump, the principal axis can be calculated by the Principal Component Analysis (PCA) algorithm. Sample results of reconstructed mesh and the detected height bumps will be shown in Sec. 7.4.

6.2.2 Discontinuity scan

To catch discontinuities over the surface of the cloth, the resolution of the Kinect sensor is not enough. Therefore, we employ diffuse reflection [Zhang *et al.*, 1999]. In Eq. (6.4), we have modeled the scenario that the detected height bumps from the curvature scan are considered as the hypothesis for the discontinuity scan. To generate an efficient ironing plan, we only care about the discontinuity on the surface, which is robust enough to find the permanent wrinkles. Therefore, we do not have to go through a calibration of the locations of the light sources to reconstruct the surface from the shading.

When using multiple light sources for the reconstruction, calibration of lighting condition is very important. In terms of the distance from the light source, the decay of illumination may result in missing of a wrinkle detection. Most of our experimental cloth or garments are made from cotton or natural fabrics, therefore, we can assume the luminance reflected from the surface is isotropic. For generality, we employ a Lambertian reflectance model [Oren and Nayar, 1995] to formulate the surface luminance. The Lambertian model can be written as,

$$I(u,v) = a(u,v) \cdot \boldsymbol{n}(u,v) \cdot \boldsymbol{s}(u,v) \tag{6.12}$$

where for a pixel location (u,v), I is the intensity of the diffusely reflected light, a is the albedo, $\boldsymbol{n}$ is the surface normal, and $\boldsymbol{s}$ is the intensity of incoming lights. For two different light sources, a will be the same, and $\boldsymbol{n}$, $\boldsymbol{s}$ are different. To catch the discontinuity in all directions, two near orthogonal light sources are used.

We assume the environmental illumination is negligible when the indoor lights are turned off. From the camera capture, the current view can be captured as I_{ref1} and I_{ref2} using the first and the second light sources. With a new piece of cloth, we turn the first and the second light source on, independently, and record the new camera views as I_1 and I_2. Given the previous camera observation, a reflection look-up table can be established. Then the calibrated value of I'_1 and I'_2 can be calculated as,

$$I'_1(u,v) = I_1(u,v)/I_{ref1}(u,v) \tag{6.13}$$

$$I'_2(u,v) = I_2(u,v)/I_{ref2}(u,v) \tag{6.14}$$

Figure 6.4 shows a comparison of the surface without and with Lambertian model correction. We can see that with the Lambertian model correction, the surface of a piece of cloth is not subject

to the change of illumination condition. The final output image is an combination of images from two light sources by $I = \sqrt{I_1'^2 + I_2'^2}$.

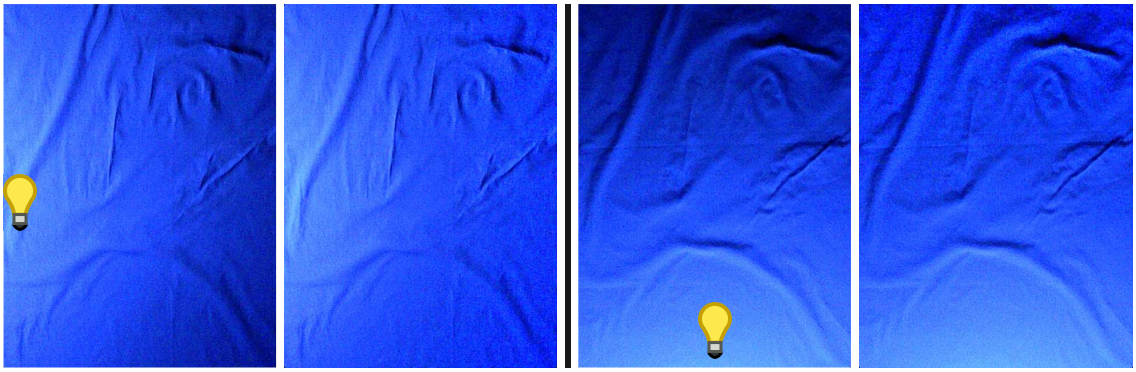


Figure 6.4: In each group (left two images or right two images), from left to right are original image and image with Lambertian correction. Two samples are shown here. One is with light source on the left and the other is at the bottom.

With the normalized surface image, we run the previous trained SIFT-SVM classifier on each pixel and classify them into permanent wrinkle pixels by the confidence score. Given all wrinkle pixels, in terms of their connectivity, the Hough transform is applied to find line segments as the detected permanent wrinkles. Non-Maximum Suppression is used for avoiding multiple wrinkles at one location. Sample results are shown in Sec. 7.4.

6.3 Ironing Procedure

6.3.1 Position-based control vs. force-based control

Force-based control usually requires a precise force sensor to generate real-time feedback, which is another cost. Empirically, the ironing task does not require very accurate force feedback to control the path. Therefore, for the ironing task, we start with an alternative method — position control. We place a foam (about 6 cm thick) under the ironing target, with which position-based control can gain similar press effects of force-based control. Essentially, the foam will passively response to the depth of the iron press and provide a spring-like force feedback; the force will be a linear function of the depth in a simple spring model. In addition, the position-based control is easier and more

convenient for deploying such ironing functionality on a household robot, whereas the force-based control may require a more sophisticated system. Experimental results (See sec. 7.4) show that the position-based control is able to remove many of the permanent wrinkles.

6.3.2 Ironing path planning

Ironing orientation Most irons are equipped with a V shape head which can easily push the air out from under the wrinkle and flatten the surface. In our scenario, for each targeted wrinkle, the principal axis is calculated. Then the end effector of an arm will align the head of the iron with the axis, given that the robot has previously stored the orientation of the iron to its base.

Ironing motion Given the targeted permanent wrinkle, the start and end ironing position can be generated in terms of the center of the iron base. Since we use position control for the ironing process, the height of the iron is strictly controlled by the robot arm. As mentioned before, we place a foam under the ironing target, so that when the iron is placed lower than the surface of the target, pressure will be created.

We also define two ironing motions for all cases, which are static ironing and dynamic ironing. The length of the wrinkle is transformed back to the world coordinate system, and one of the motions is selected based upon the length. More specifically, if the length of the wrinkle is less than 70% of the longer axis of the iron, the static ironing motion is selected, otherwise, the sliding motion. However, as the iron slides on the cloth, it may cause an effect of air accumulation, which may create more permanent wrinkles and height bumps. Therefore, we limit the length of the detected wrinkle in the Hough transform into at most twice the length of an iron, and break down a longer one into shorter ones.

Ironing path optimization During the final step of surface analysis, a set of permanent wrinkles $\mathcal{W}$ will be detected, all of which will be ironed. In our scenario, each permanent wrinkle is represented by a line segment. For each of the line segment, the iron will move from one end point to the other end point as an iron action. If a line segment is selected randomly at each time, the whole process could take longer time because the iron may move back and forth. Therefore, we design an algorithm that minimize such time using a greedy search. The first selected permanent wrinkle is the one with the highest probability calculated from our framework. Each time, the iron looks for the closest end point of another line segment for the next ironing process. Details of the

algorithm are described in Algorithm 4.

Algorithm 4: Wrinkle Sorting

Input: $\mathcal{W}$: a set of line segments with probability $w.prob$, and two end points $w.e1$, $w.e2$. (the ironing is from e1 to e2);

Output: $\mathcal{W}'$: re-ordered set of line segments;

$w_c \leftarrow$ Max($\mathcal{W}.prob$);
while $\mathcal{W} \neq \phi$ **do**
  $\mathcal{W}' \leftarrow \mathcal{W}' + w_c$;
  $\mathcal{W} \leftarrow \mathcal{W} - w_c$;
  $w_s \leftarrow$ Min ($w_c.e2$, $\mathcal{W}.e1 + \mathcal{W}.e2$);
  if $|w_s.e1 - w_c.e2| > |w_s.e2 - w_c.e2|$ **then**
    swap($w_s.e1$, $w_s.e2$);
    // Keep e1 as the starting iron end point.
  $w_c \leftarrow w_s$;
return $\mathcal{W}'$;

6.4 Experimental Results

To evaluate our results, we tested our method on several different garments and pieces of cloth for multiple trials. A high resolution video of our experimental results is online at http://www.cs.columbia.edu/~yli/laundry_robot.

6.4.1 Robot Setup

In our experiments, we use a Baxter research robot, which is equipped with two arms with seven degrees of freedom. We mount a Kinect sensor [Zhang, 2012] on top of the table, facing down, which has been calibrated to the robot base frame, as shown in Figure 6.3, left. As described in Sec. 6.2.1, a two-axis module is attached to the Kinect to create enough parallax for depth fusion.

6.4.2 Curvature scan

The curvature scan aims at detecting large height bumps over the cloth surface. For each scan, we start the two motors simultaneously for two completed rotations. The quality of the reconstructed mesh is gradually improved as more scans are taken. Figure 6.5 shows two samples of height bump detection using our curvature scan. As described in Sec. 6.2.1, the height bumps are detected by computing the Hessian matrix and volume estimation. The Middle image in each row is the detected height bumps, which are rendered in grey-scale color (the higher, the darker). In terms of the bump volume, we discard small ones with a given threshold.

The results of GMM are shown in the right images. We relax the covariance of each Gaussian model in GMM to give a higher probability to the area between adjacent height bumps. Therefore, we can see that when several height bumps are close to each other, the fitted Gaussian model tends to be joined. Those heatmaps in Figure 6.5 can be considered as a hypothesis for the detection of the permanent wrinkle.

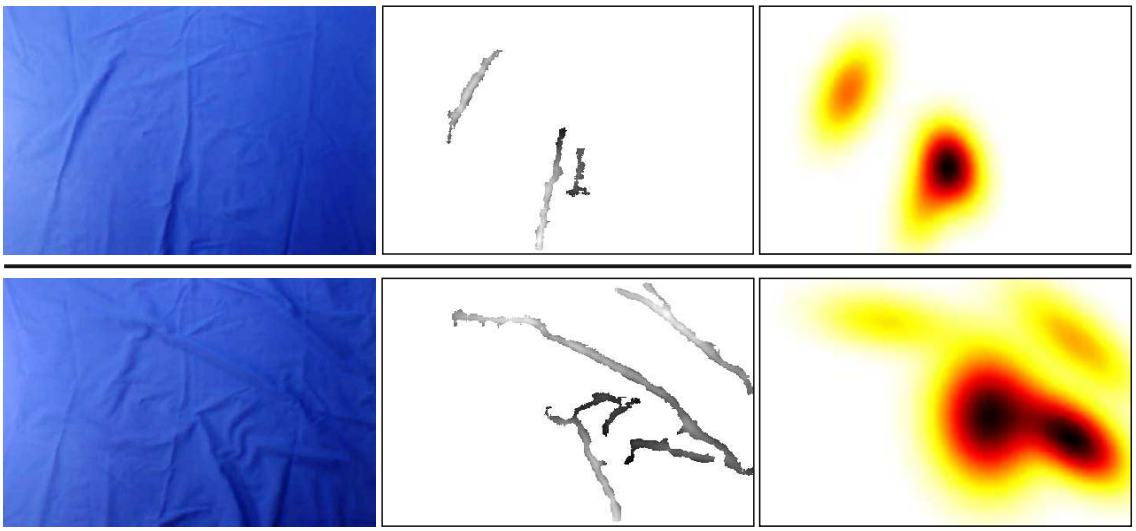


Figure 6.5: Examples of height bump detection and Gaussian Mixture Model (GMM) approximation. From left to right of each row are: reconstructed surface (rendered with color), detected height bumps (small ones are discarded), and GMM approximation. TOP ROW: 3 separated height bumps are detected. BOTTOM ROW: 4 separated height bumps are detected.

6.4.3 Discontinuity scan

The discontinuity scan aims at detecting shadow discontinuities over the cloth surface. We manually create a training dataset which contains thousands of pixels on the discontinuity region by manually labeling. Then we apply the trained SVM classifier on new images. Figure 6.6 shows an example of discontinuity detection. The middle image is the probability map from the SVM classification. With the Hough line detection, we obtain a set of line segments for ironing paths. Note that since we limit the length of each line segment for stable ironing purposes, longer line segments may be broken down into smaller pieces, which cannot be seen from the image.

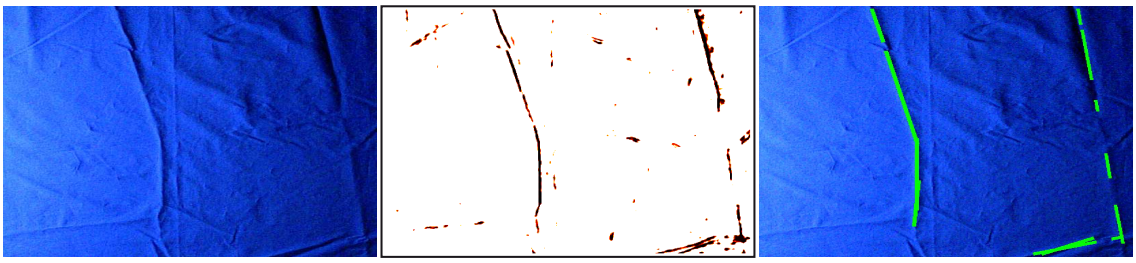


Figure 6.6: Example of the discontinuity scan. From left to right are original image, SVM confidence score map, and permanent wrinkle detection results overlaid on the original image. Each wrinkle is represented as a line segment in green color.

6.4.4 Robotic ironing

We have tested our approach on several different pieces of cloth and garments. Here we demonstrate two examples of all the steps in robotic ironing in Figure 6.7 and Figure 6.8. It starts with the curvature scan to detect height bumps using the Kinect sensor. The detected bumps are shown in (B). Then we apply GMM approximation on (B) given the number of bumps as shown in (C). With discontinuity scan (D) and our probabilistic multi-sensor model, permanent wrinkles are detected as shown in (E) in green line segments. Those wrinkles, represented as line segments, are sorted in a way that robot can iron them with minimum movement and alignment. (G) shows an image of the cloth after ironing, and (H) is after manually flattening (note that permanent wrinkles cannot be removed by such flattening). We can see that some discontinuities on or close to the height bumps can be removed in the final permanent wrinkle detection.

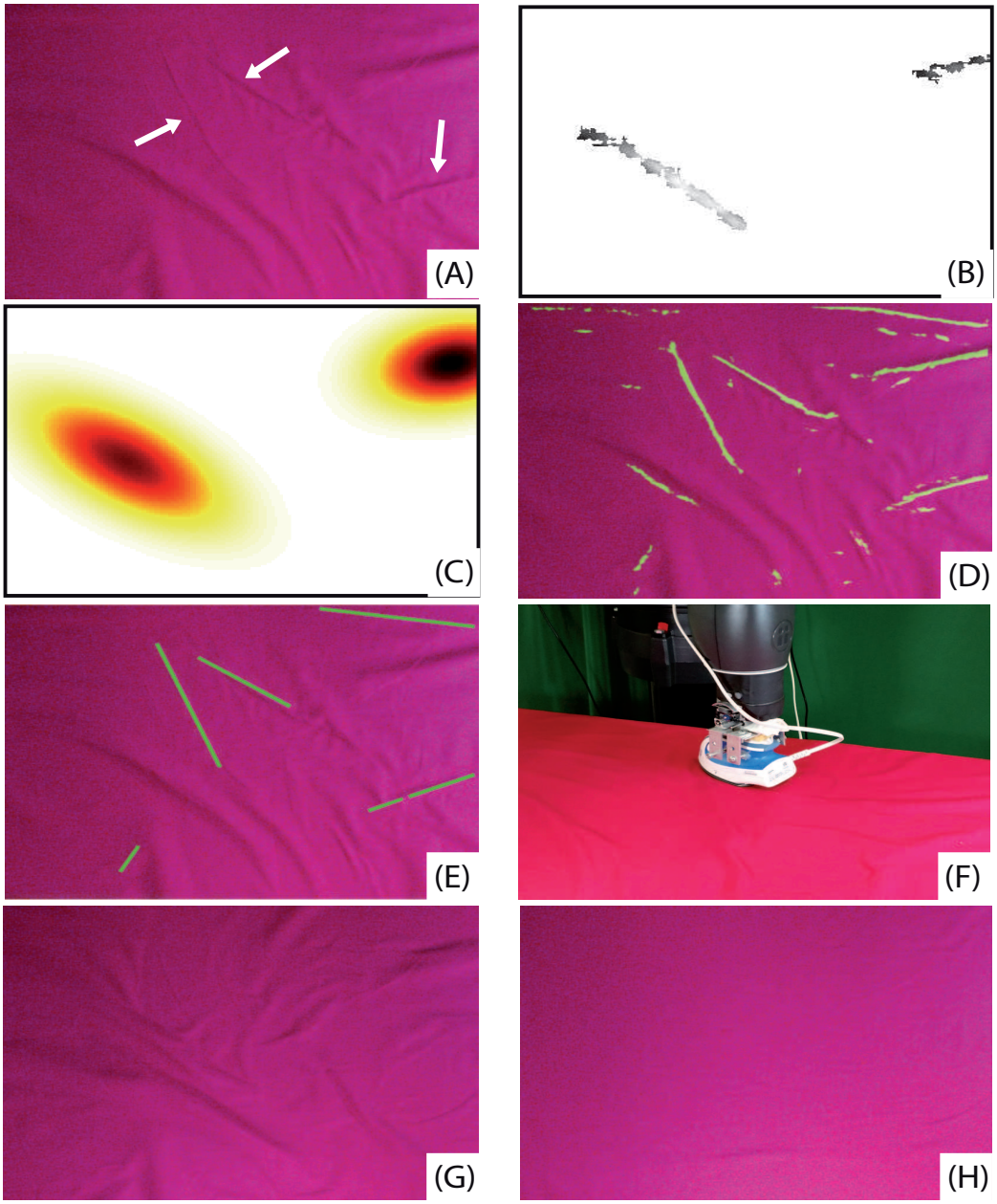


Figure 6.7: Best viewed by zooming. one example of whole process of the robotic ironing a red cloth. (A): original roughly flattened cloth (white arrows show some obvious wrinkles). (B): Detected height bumps. (C): GMM fitting results. (D): SVM-based discontinuity detection results. (E): Final permanent wrinkle detection results, which combines result from C and D. (F): The Baxter robot ironing. (G): The ironing result. (H): The ironing result after manually flattening.

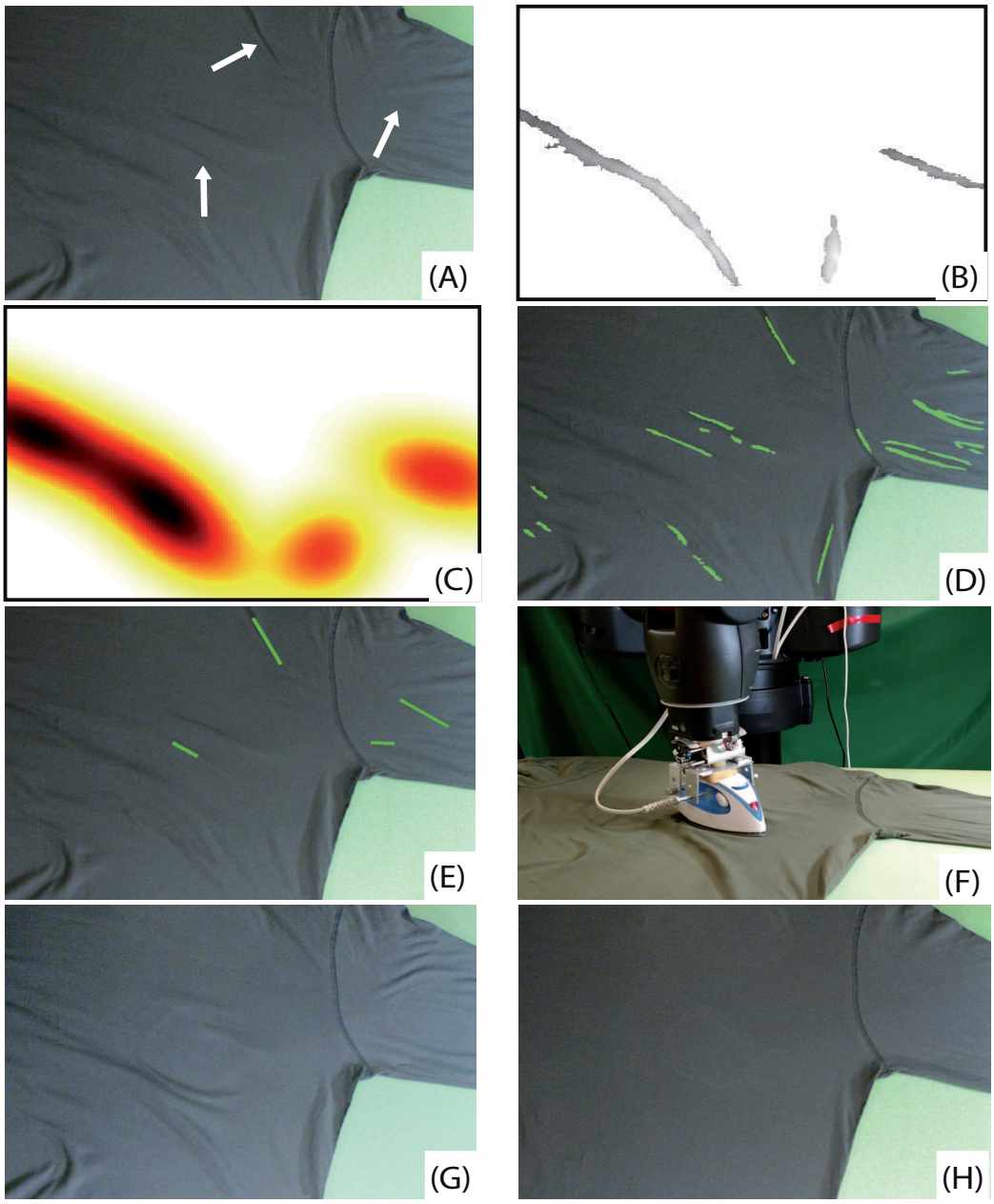


Figure 6.8: Best viewed by zooming. One example of whole process of the robotic ironing a green long-sleeve sweater. (A): original roughly flattened cloth (white arrows show some obvious wrinkles). (B): Detected height bumps. (C): GMM fitting results. (D): SVM-based discontinuity detection results. (E): Final permanent wrinkle detection results, which combines result from C and D. (F): The Baxter robot ironing. (G): The ironing result. (H): The ironing result after manually flattening.

We also show two cases in Figure 6.9, where the subjects are more complicated garments. Those garments usually have some decorations or non-smooth patterns (seams) over the surface, which may be classified as permanent wrinkles. Ironing these mis-classified wrinkles usually will not bring more permanent wrinkles but is time consuming. In Figure 6.9, from left to right rows are the object before ironing, detected permanent wrinkles overlaid (blue line segments), and the ironing results after manually flattening. The top row is a pair of pants. Some of the seams on the pants are detected as the permanent wrinkles (blue line segments). The border of the pants may cause illumination discontinuities, which lead to mis-classification of the permanent wrinkle. One way to handle this case is that using a pre-computed mask by thresholding the color image, then use it to filter the region of interest. The bottom row shows a pink sweater, which is soft compared to other garments. Because of a seam across the chest part (appears vertically in the image), there are lots of wrinkles which are perpendicular to the seam. Some of these are detected as the permanent wrinkles. Also, a small part of the garment border is detected as a permanent wrinkle.

6.4.5 Quantitative ironing

In order to evaluate how effective our robotic ironing is on removing permanent wrinkles, we conduct a research of removing known permanent wrinkles that are created manually. The subjects are a piece of blue cloth (see Figure 6.10), a piece of red cloth (see Figure 6.7), and a green sweater (see Figure 6.8). Given a flattened piece of cloth or a garment, we used an iron to create some permanent wrinkles, which cannot be removed by pulling the boundaries. We designed 6 experiments with various numbers of wrinkles with different poses and sizes. Table 6.1 shows the statistical results of the quantitative ironing. The five columns from left to right are: number of the experiments, number of the wrinkles we have created, number of the detected wrinkles using our algorithm, number of the detected wrinkles after ironing, and number of wrinkle pixels after ironing over number of wrinkle pixels before ironing detected by our algorithm. The last column shows how many percentages of the wrinkles can be removed after ironing in terms of pixels. Here we use the same wrinkle detection algorithm to check if any permanent wrinkle left. Specifically, $\|W\|$ represents how many wrinkle pixels before the ironing, and $\|W_{new}\|$ represents how many wrinkle pixels after ironing. We can see that, in most cases, the robotic ironing can effectively remove the permanent wrinkles, or reduce the pixels of the permanent wrinkles on a piece of cloth. For the two wrinkles that still

Figure 6.9: Best viewed by zooming. The permanent wrinkles are marked as blue line segments. TOP ROW: One example of ironing a pair of pants. Some of the seams on the pants are detected as the permanent wrinkles. BOTTOM ROW: One example of ironing a long sleeve pink sweater. Because of a seam across the chest part (appears vertically in the image), there are lots of wrinkles which are perpendicular to it. Some of these are detected as the permanent wrinkles. Also, a small part of the garment border is detected as a permanent wrinkle. For each row, from left to right are the object before ironing, detected permanent wrinkles overlaid, and the ironing results after manually flattening.

remain, one of them is because the ironing time was not enough and the discontinuity is not fully smoothed. The other one is because our algorithm did not detect the permanent wrinkle, which may be due to the threshold of the confidence score of our SVM classifier.

Exp. #	# of W	# of detected W	# of W after ironing	$\|W_{new}\|/\|W\|$
Blue cloth	4	4	0	0/960
Blue cloth	5	4	1	151/680
Blue cloth	3	3	0	0/591
Red cloth	5	5	0	0/1013
Green sweater	3	3	1	212/751
Green sweater	4	4	0	0/632

Table 6.1: W stands for wrinkle. The five columns from left to right are: candidate garments of the experiments, number of the wrinkles we have created, number of the detected wrinkles using our algorithm, number of the detected wrinkles after ironing, and number of wrinkle pixels after ironing and flattening over number of wrinkle pixels before ironing detected by our algorithm. In most cases, the robotic ironing can effectively remove the permanent wrinkles, or reduce the pixels of the permanent wrinkles on a piece of cloth. For the two wrinkles that still remain, one of them is because the ironing time was not long enough. The other one is because our algorithm did not detect the permanent wrinkle.

6.4.6 Ironing with optimized path

We have tested our ironing path plan on several different garments with more than one permanent wrinkle detected. As described in Alg. 4, when multiple permanent wrinkles are detected, an optimized ironing path is generated, and the iron will traverse all the wrinkles. An optimized path will reduce the unnecessary alignment and movement of the iron, as well as inverse-kinematics computation.

Figure 6.10 shows an example of the ironing with an optimized path. (A): A piece of cloth contains several height bumps and permanent wrinkles. Green line segments are detected permanent wrinkles. White circle is the starting position and white arrows show the path orientation. (B): Results after ironing and manually flattening. (C)-(H): Key frames of the whole ironing process in an order. The iron first aligns with the first wrinkle and moves towards the starting position. After each ironing, the iron will be lifted up about $5cm$, and move to the next permanent wrinkle. In this case, the whole ironing process takes about 35 seconds, which is faster than a path without optimization or a randomized path.

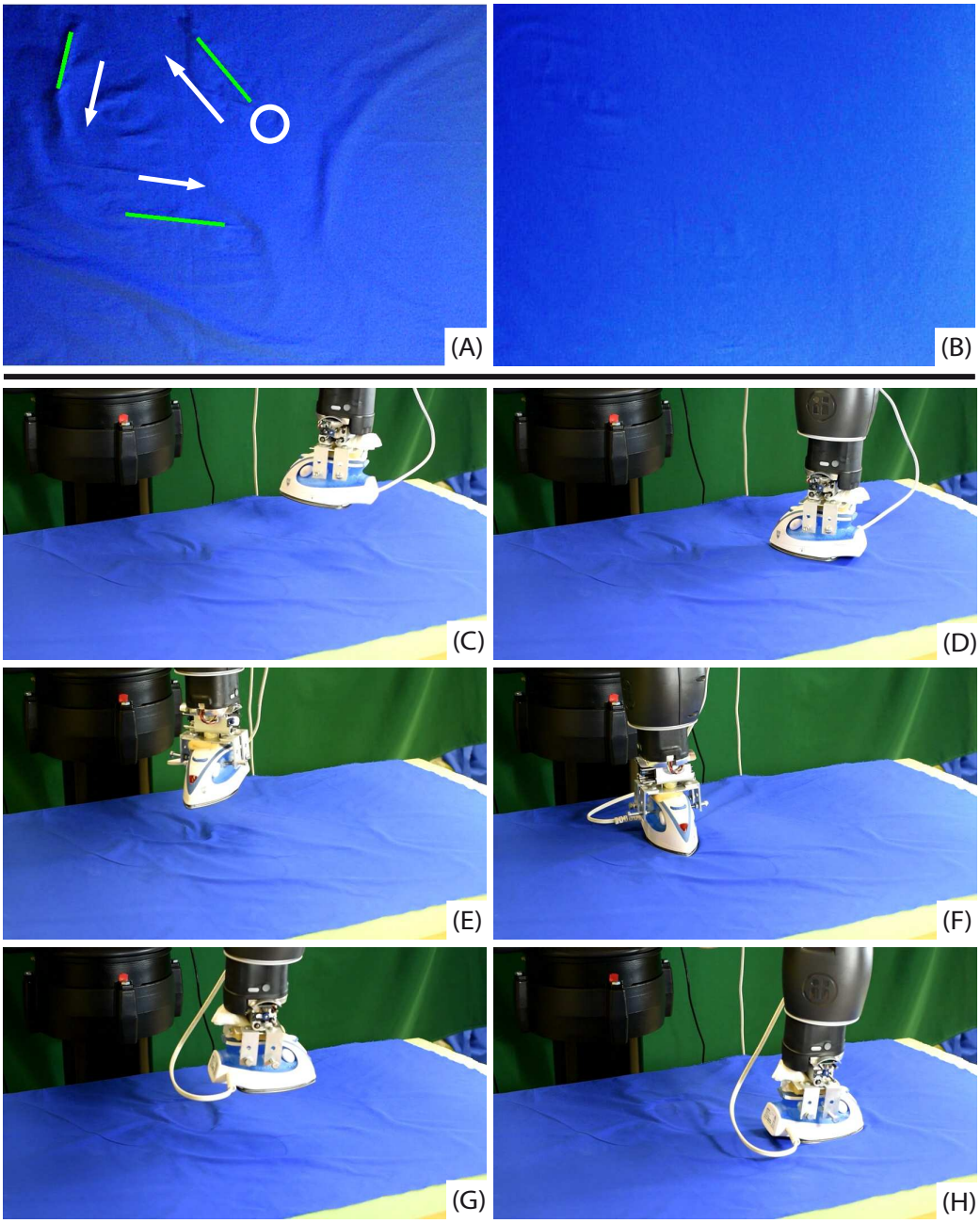


Figure 6.10: An example of robotic ironing with an optimized path. In this case, three permanent wrinkles are detected. (A): Green line segments are detected permanent wrinkles. White circle is the starting position and white arrows show the path orientation. (B): Results after ironing and manually flattening. (C)-(H): Key frames of the whole ironing process in an order.

6.5 Exploration of static ironing

We found that with different ironing pressure, the permanent wrinkle can be reduced, removed, or more wrinkles and height bumps can be created. Therefore, we design a study on how the pressure of the iron related to the permanent wrinkle size. We first manually create three permanent wrinkles with different size: small, medium, and large, as shown in Figure 6.11, second row. Then the robot is set to use three different heights to iron the wrinkles (note that we use position control to create pressure). Each image column represents a pressure level using static ironing motion. From left to right are light, medium, and heavy. We finally applied our probabilistic multi-sensor framework on the ironed result to see if the surface is wrinkle-free, as shown in Figure 6.11, pink color. The percentage number in the bottom-right corner indicates the percentage of the detected permanent wrinkle pixels over the whole image. For each trial, we repeated 10 times and the results are relatively consistent.

From the results, we can see that for all size of the wrinkles, the medium pressure is practical for most of the cases. The light pressure may not be able to flatten the wrinkle whereas the heavy pressure sometimes create extra wrinkle or height bumps around the iron shape. We can also see that our probabilistic multi-sensor framework is able to capture most permanent wrinkles and ignore those height bumps which can be flattened by iterative pulling.

6.6 Summary

In this chapter, we have presented a probabilistic multi-sensor framework for wrinkle detection in a robotic ironing task. We analyze the surface by first using a curvature scan by a Kinect sensor to find height bumps, which should not be included in the ironing. Then by using a discontinuity scan with two light sources, and a SVM based classifier to find discontinuities on the surface. Combining results from the two scans, permanent wrinkles can be detected and represented as line segments. A Baxter robot, holding an iron in one hand, irons the surface guided by those detected line segments in an optimized order. Experimental results show that with our detected wrinkles, the Baxter robot is able to iron and remove them iteratively and produce a wrinkle-free area on the cloth.

There are some failure cases where the permanent wrinkles are not fully removed. Based on our experimental results, the reasons can be summarized as the following. First, our permanent

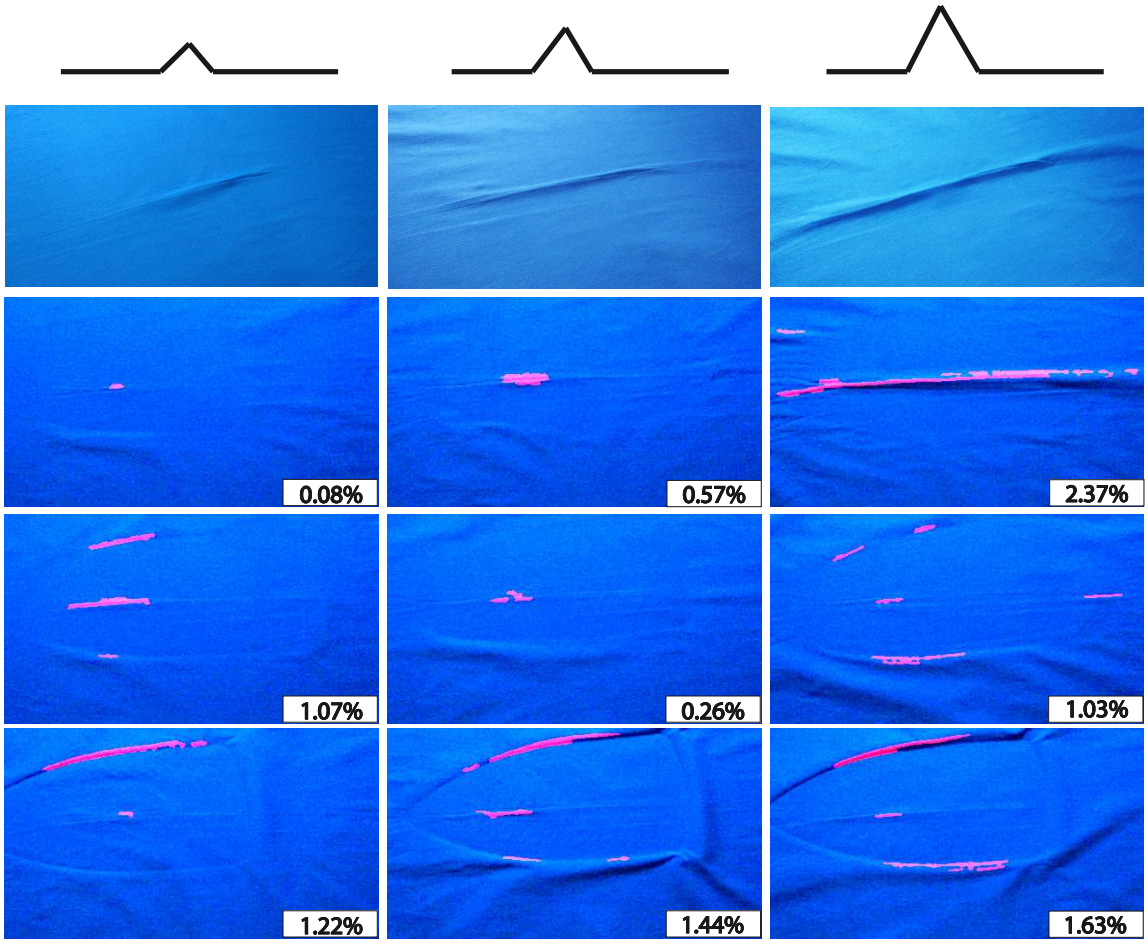


Figure 6.11: Examples of static ironing study. The top row of each column shows a sketch of three wrinkles with different heights (from left to right: low, medium, high). The second row shows images of the original permanent wrinkles. Then the images from the third row to the last row are ironed by pressure level in light, medium, heavy, respectively. Pink regions are detected permanent wrinkles. The number in the bottom-right corner shows the percentage of wrinkle pixels over the whole image.

wrinkle detection algorithm does not catch those wrinkles. With more wrinkle instances added to the training set, the detection accuracy can be improved. In addition, the ironing time is not long enough to remove the wrinkle. Also, a better force control system may also help. To my experience, we even need to spray some water on the wrinkle area and then iron it, which may produce better results. But this requires a more complicated system. Another issue is when ironing some permanent

wrinkles, it is possible to produce more permanent wrinkles if the targeted wrinkle is relatively big. This could be solved by pulling the garment border in a direction perpendicular to the principal direction of the permanent wrinkle.

Chapter 7

Folding

Robotic folding of a garment is a difficult task because it requires sequential manipulations of a highly unconstrained, deformable object. Given the garment shape, the robot can fold it by following a folding plan [Miller *et al.*, 2011][Miller *et al.*, 2012]. However, the layout of the same folding action can vary in terms of the material properties such as cloth hardness and the environment such as friction between the garment and the table. Given the starting and ending folding positions, different folding trajectories will lead to different results. In this chapter, we propose a novel method that learns optimal folding trajectory parameters from predicted thin shell simulations of similar garments, which can then be applied to a real garment folding task (see Figure 7.1). The contributions are:

- A fast and robust algorithm that can detect garment key points such as sleeve ends, collar, and waist corner, automatically. These key points can be used for folding plan generation.
- An online optimization algorithm that learns optimal trajectories for manipulation from mathematical model evolution combined with predictive thin shell simulation.
- A novel approach that adjusts the simulation environment to the robot working environment for the purpose of creating a similar manipulation result.
- The trajectories are general in that they can be scaled to accommodate similar garments of different size.

- Experimental results with a Baxter dual-arm robot showing successful folding trajectories for a number of different garments including sweaters, pants, and towels.

Figure 7.2 shows the complete pipeline of garment manipulation, as well as the key steps of garment folding. The garment folding is the final step of the entire pipeline of garment manipulation which contains grasping, visual recognition, regrasping, unfolding, placing flat, and folding. Our previous work [Li *et al.*, 2014a][Li *et al.*, 2014b][Li *et al.*, 2015a] has successfully addressed all the stages of the pipeline with the exception of the final folding task. This chapter specifically addresses the robotic folding task (purple rectangle in Figure 7.2) with the goal of finding optimal trajectories to successfully fold garments.

We begin by assuming the garment is placed flat on the table initially, as shown in our previous work [Li *et al.*, 2015a]. By detecting the key points of the garment (see section 7.3.1), a pre-defined folding plan is used to create optimal trajectories for folding the garment. After several steps, we obtain a desired folding result in the real world using the Baxter robot, which is comparable to the result from the simulation.

7.1 Simulation Environment

7.1.1 Folding Pipeline in Simulation

In the model simulation, we use a physics engine [Maya, 2014] to simulate the movement and deformation of the garment mesh models. We assume there is only one garment for each folding task, which has been placed flat on a table. A virtual table is added to the scene which the garment lies on, as shown in Figure 7.1, top.

During each folding step, the robot arm picks up a small part of the mesh, moves it to the target position following a computed trajectory, and places it on the table to simulate an entire folding scenario. If the part of the garment to be folded is relatively wide, then both left and right arms may be involved. The trajectory is generated using a Bézier curve, which will be discussed in section 7.2.

Most of the garment mesh models are built from our test garments. A garment mesh is created by first extracting the contour of the garment (see section 7.3.1). Then by inserting points on the inside of the garment contour, we triangulate a mesh by connecting these points. Lastly, we mirror the mesh to construct a two-sided garment mesh (see figure 7.3).

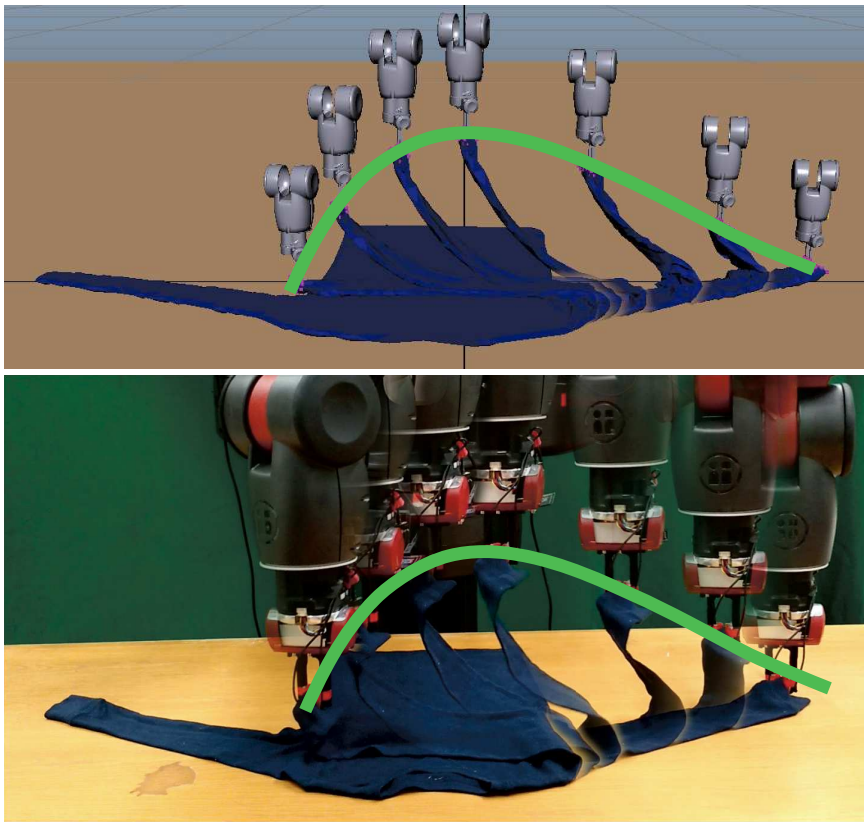


Figure 7.1: Comparison of our simulation of robotic manipulation (TOP) and real robot implementation (BOTTOM). The green curves show the virtual and the real trajectories for folding.

7.1.2 Parameter Adaptation

There are two key parameters needed to accurately simulate the real world folding environment. The first is the material properties of the fabric, and the second is the frictional forces between the garment and the table.

Material properties Through many experiments, we found that the most important property for the garments in the simulation environment is shear resistance. It specifies the amount the simulated mesh model resists shear under strain; when the garment is picked up and hung by gravity, the total length will be elongated due to the balance between gravity force and shear resistance. An appropriate shear resistance measure allows the simulated mesh to reproduce the same elongation as the real garment. This parameter will bridge the gap between the simulation and the real world

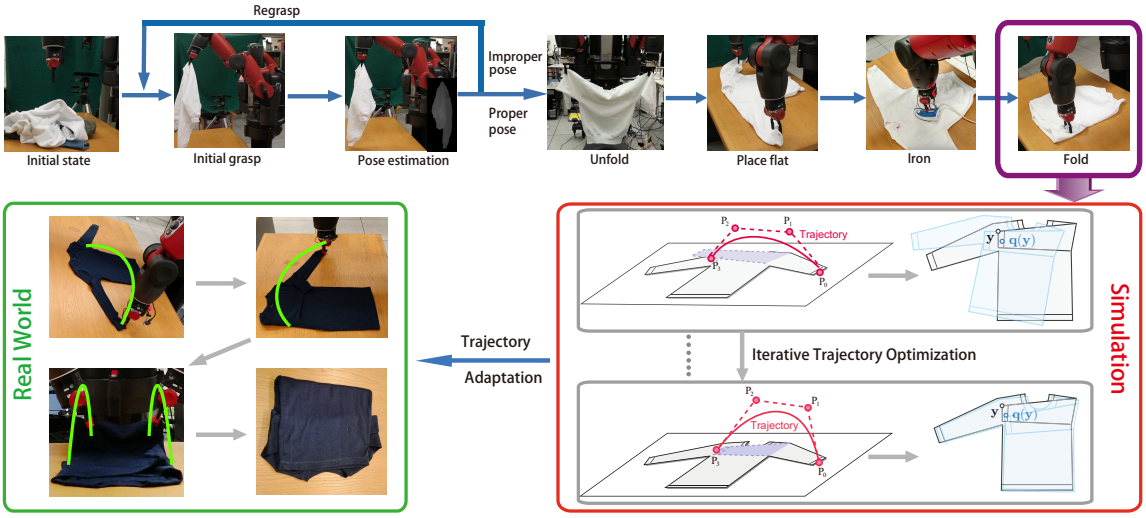


Figure 7.2: TOP ROW: The entire pipeline of dexterous manipulation of deformable objects. In this chapter, we are focusing on the phases of garment folding, as highlighted in the purple rectangle. BOTTOM ROW: Details of the folding procedure. We apply off-line simulation with iterative trajectory optimization to find the best trajectory for a specific folding action by comparing the result (light blue contour) with template (black contour). Similar steps are repeated until the garment is folded in the simulator. Then all the folding trajectories are exported, adapted, and implemented on a real robot. Green arcs illustrate the actual trajectories of robotic arms.

for the garment mesh model.

For each new garment, we follow the steps described below to measure the shear resistance. Figure 7.4 shows an example.

- Manually pick one extremum part of the garment such as the sleeve end of a T-shirt, the waist part of a pair of pants, and a corner of a towel.

- Hang the garment under gravity and measure the length between the picking point and the lowest point as L_1

- Slowly put down the garment on a table and keep the picking point and the lowest point in the previous step at maximum spread condition. Measure the distance between these two points again as L_2. The shear resistance fraction is defined as the following

$$shear\_frac = (L_1 - L_2)/L_2 \tag{7.1}$$

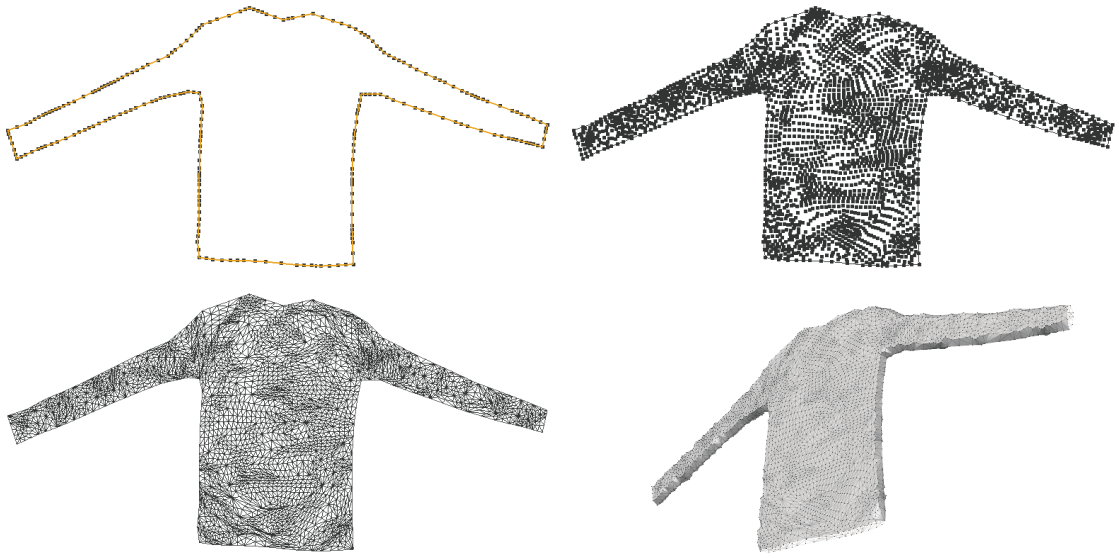


Figure 7.3: Garment models. TOP LEFT: the input contour. TOP RIGHT: we insert vertices into the internal region. BOTTOM LEFT: we build a flat triangle mesh using the contour and the inserted vertices. BOTTOM RIGHT: we shift the contour vertices and mirror the mesh to create the garment mesh.

- We then pick up and hang the virtual garment in *Maya*, adjusting the *Maya* shear parameter such that the shear fraction as calculated in the simulator is identical to the real world.

Frictional forces The surface of the table can be rough if covered by a cloth sheet or slippery if not covered, which leads to variance in friction between the table and garment. A shift of the garment during the folding can possibly impair the whole process and cause additional repositioning. Adjusting the frictional level in the simulation environment to the real world is crucial and necessary for trajectory optimization.

To measure the friction between the table and the garment, we do the following steps.

- Place a real garment on the real table of length L_t.

- Slowly lift up one side of the real table, until the garment in the real world begins to slide. The lifted height is H_s. The friction angle is computed as,

$$\angle_{Friction} = \sin^{-1}(H_s/L_t) \tag{7.2}$$

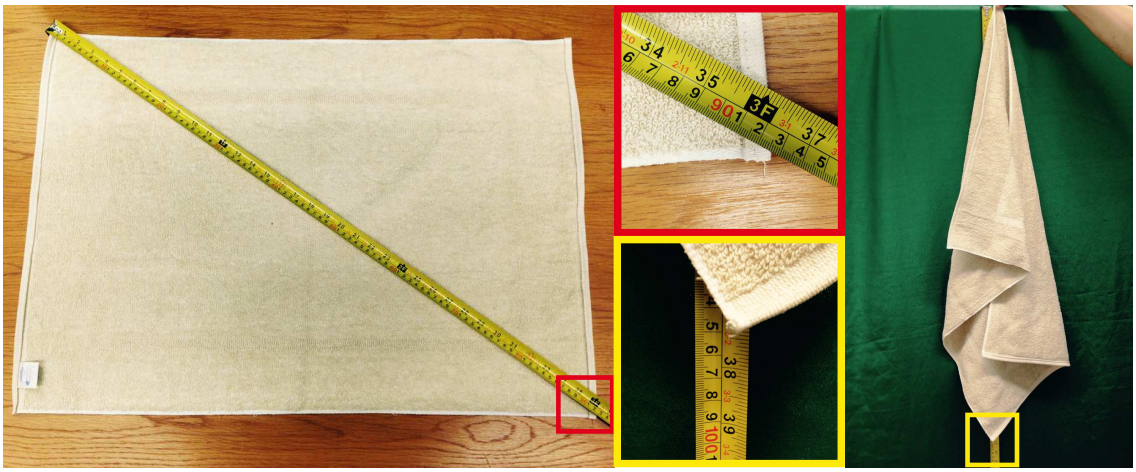


Figure 7.4: Method for measuring the shear resistance. LEFT: Diagonal length measurement. MIDDLE: Zoomed in regions. RIGHT: The garment is hanging under gravity.

- In the virtual environment, the garment is placed flat on a table with gravity. Assign a relatively high friction value to the virtual table. Lift up one side of the virtual table to the angle of $\angle_{Friction}$.

- Gradually decrease the frictional force in the virtual environment until the garment begins to slide. Use this frictional force in the virtual environment as it mirrors the real world

With these two parameters set up, we obtain similar manipulation results for both the simulation and the real garment.

7.2 Trajectory Optimization

The goal of the folding task is specified by the initial and folded shapes of the garment, and by the starting and target positions of the grasp point (as in Figure 7.5). Given the simulation parameters, we seek the trajectory that effects the desired set of folds. We first describe how to optimize the trajectory for a single end effector, and then discuss the case of two end effectors.

7.2.1 Trajectory parametrization

We use a Bézier curve [Farin, 1988] to describe the trajectory. An n-th order Bézier curve $\mathbf{T}(u)$ has $(n+1)$ control points $\mathbf{P}_k = (P_{k,x}, P_{k,y}, P_{k,z})^T \in \mathbb{R}^3$, defined by

$$\mathbf{T}(u) = \sum_{k=0}^{n} B_k^n(u)\mathbf{P}_k, \tag{7.3}$$

where $B_k^n(u) = \binom{n}{k}(1-u)^{n-k}u^k$ are the Bernstein basis functions.

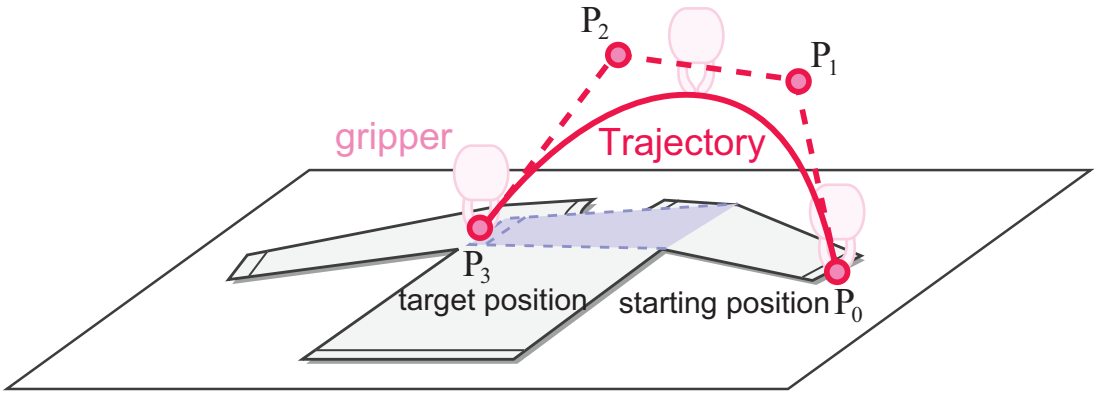


Figure 7.5: An example of the folding task: we want to fold a sleeve into the blue target position, by using a robotic gripper to move the tip of the sleeve (grasp point) from the starting position ($\mathbf{P}_0$) to the target position ($\mathbf{P}_3$), following a trajectory, shown as the red curve. $\mathbf{P}_1$ and $\mathbf{P}_2$ are knot points that form the Bézier trapezoid.

We use $n = 3$ for simplicity, but our method can be easily extended to deal with higher order curves. $\mathbf{P}_0$ and $\mathbf{P}_3$ are fixed to the specified starting and target positions of the grasp point (as in Figure 7.5). The intermediate control points $\mathbf{x} = (\mathbf{P}_1^T, \mathbf{P}_2^T)^T$ can then be adjusted to define a new trajectory using the objective function defined below. The update rule is described in section 7.2.2.

$$\mathbf{x}_{opt} = \underset{\mathbf{x}}{\operatorname{argmin}} \{\underbrace{l_{\mathbf{x}} + \alpha D(\mathcal{S}_t, \mathcal{S}_{\mathbf{x}})}_{C(\mathbf{x})}\}^2. \tag{7.4}$$

Here $C(\mathbf{x})$ is a cost function with two terms. The first term penalizes the trajectory length $l_{\mathbf{x}}$, thus preferring a folding path that is efficient in time and energy. The second term seeks the desired fold, by penalizing dissimilarity $D(\mathcal{S}_t, \mathcal{S}_{\mathbf{x}})$ between the desired folded shape $\mathcal{S}_t$, compared to the

shape $\mathcal{S}_\mathbf{x}$ obtained by the candidate folding trajectory $\mathbf{x}$, as predicted by a cloth simulation; we used a physical simulation engine [Maya, 2014], for the cloth simulation. The weight α balances the two terms; we used $\alpha = 10^3$ in our experiment. See section 7.2.2 for optimization details.

Intuitively, dissimilarity measures the difference between the desired folded shape and the folded garment in simulation. We define the dissimilarity term as

$$D(\mathcal{S}_t, \mathcal{S}_\mathbf{x}) = \frac{1}{|\mathcal{S}_t|} \int_{\mathcal{S}_t} \|\mathbf{q}(\mathbf{y}) - \mathbf{y}\| dA, \tag{7.5}$$

where $|\mathcal{S}_t|$ is the total surface area of the garment mesh including both sides of the garment, $\mathbf{y} \in \mathcal{S}_t$ is a point on the target folded shape $\mathcal{S}_t$, $\mathbf{q}(\mathbf{y}) \in \mathcal{S}_\mathbf{x}$ is the corresponding point on the simulated folded shape, and dA is the area measure, see Figure 7.6, left. Our implementation assumes $\mathcal{S}_t$ and $\mathcal{S}_\mathbf{x}$ are given as triangle meshes, and discretizes (7.5) as

$$\tilde{D}(\mathcal{S}_t, \mathcal{S}_\mathbf{x}) = \frac{1}{|\mathcal{S}_t|} \sum_i \|\mathbf{q}_i - \mathbf{y}_i\| A_i, \tag{7.6}$$

where $\mathbf{y}_i$ is the barycenter of i-th triangle on the target shape, $\mathbf{q}_i$ is the (corresponding) barycenter of i-th triangle on the simulated shape, and A_i is the area of the i-th triangle on the target shape. In Figure 7.6, right, the barycentric dual area A_i associated with this vertex $\mathbf{y}_i$ is defined as the area of the polygon created by connecting the barycenters of the triangles adjacent to $\mathbf{y}_i$.

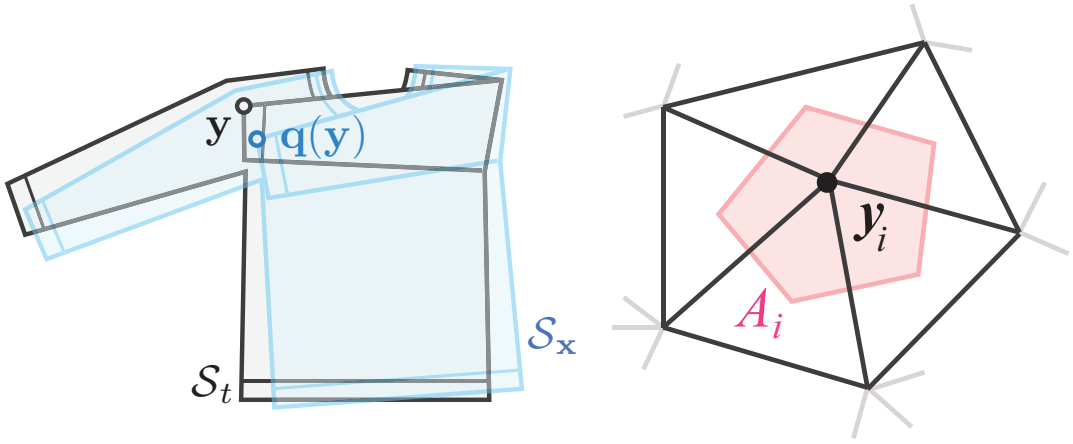


Figure 7.6: LEFT: The dissimilarity captures the misalignment between $\mathcal{S}_t$ and $\mathcal{S}_\mathbf{x}$ by integrating the distance between the corresponding points $\mathbf{y} \in \mathcal{S}_t$ and $\mathbf{q}(\mathbf{y}) \in \mathcal{S}_\mathbf{x}$ over the garment. RIGHT: The barycentric dual area A_i associated with this vertex $\mathbf{y}_i$ is defined as the area of the polygon created by connecting the barycenters of the triangles adjacent to $\mathbf{y}_i$.

To compute the trajectory length $l_{\mathbf{x}}$, we use the De Casteljau's algorithm [Farin, 1988] to recursively subdivide the Bézier curve $\mathbf{T}$ into a set of Bézier curves $\mathbf{T}^{(j)}$, until the deviation between the chord length ($\|\mathbf{P}_0^{(j)} - \mathbf{P}_3^{(j)}\|$) and the total length between the control points ($\sum_{i=0}^{2} \|\mathbf{P}_i^{(j)} - \mathbf{P}_{i+1}^{(j)}\|$) for each subdivided curve $\mathbf{T}^{(j)}$ is sufficiently small. Then, $l_{\mathbf{x}}$ is approximated by summing up the chord lengths of all the subdivided curves: $l_{\mathbf{x}} \approx \sum_j \|\mathbf{P}_0^{(j)} - \mathbf{P}_3^{(j)}\|$.

We initialize $\mathbf{P}_1$ and $\mathbf{P}_2$ as

$$\mathbf{P}_1 = \frac{2}{3}\mathbf{P}_0 + \frac{1}{3}\mathbf{P}_3 + h\|\mathbf{P}_0 - \mathbf{P}_3\|\mathbf{e}_v, \tag{7.7}$$

$$\mathbf{P}_2 = \frac{1}{3}\mathbf{P}_0 + \frac{2}{3}\mathbf{P}_3 + h\|\mathbf{P}_0 - \mathbf{P}_3\|\mathbf{e}_v, \tag{7.8}$$

where $\mathbf{e}_v$ is the unit vector in the upward vertical direction, h is a constant, which is set to $1/3$, which means the initial trajectory will have equal horizontal extent between knot points.

7.2.2 Optimization.

To optimize equation (7.4), we apply a secant version of the Levenberg-Marquardt algorithm [Madsen *et al.*, 2004][Nocedal and Wright, 2006]. For the current trajectory generated by $\mathbf{x}$, we estimate the derivative $\nabla C(\mathbf{x})$ of the cost function $C(\mathbf{x})$ numerically, by sampling slightly modified trajectories $\mathbf{x} + \delta\mathbf{e}_j$, where $\mathbf{e}_j, 1 \leq j \leq \dim(\mathbf{x})$, are the orthonormal bases, and we used $\delta = 10^{-1}$ in our implementation.

The secant version of Levenberg-Marquardt algorithm iteratively builds a local quadratic approximation of $\{C(\mathbf{x})\}^2$ based on the numerical derivative, and then takes a step toward an improved state. The direction of the step is a combination of the steepest gradient descent direction and the conjugate gradient direction. We use the specific approach described by Madsen et al. [Madsen *et al.*, 2004] (see §3.5 therein). The iterative procedure terminates when the improvement in $\{C(\mathbf{x})\}^2$ becomes sufficiently small.

7.2.3 Multiple Arms

In the case of using multiple arms, we associate an individual trajectory $\mathbf{x}_i$ to each of the arms R_i. We then extend the state variable to $\mathbf{x} = (\mathbf{x}_1^T, ...)^T$. The rest of the optimization procedure is the same as the single arm case. Note that both single and dual-arm trajectories are in 3D space. The

optimization for dual-arm trajectories is able to find a solution which will overcome failures such as shown in Figure 1.2 bottom.

7.3 Application to Garment Folding

7.3.1 Key Points Localization

We assume the garment is placed roughly in the center of the table, as shown in the Figure 7.7, bottom. Our first step is to segment the garment from the background. Since we can easily obtain a table or a surface with homogeneous color [Miller *et al.*, 2011][Stria *et al.*, 2014a], a fast color-based supervised image segmentation method is suitable for our task. We apply a marker-based watershed method [Vincent and Soille, 1991] with the four corners of the table initialized as background and the center initialized as the foreground. More complicated scenes can employ advanced image segmentation algorithms such as GrabCut [Rother *et al.*, 2004].

To register the feature points, such as the corner points of sleeves, we employ a 2D registration technique to register a pre-defined garment template (as in Figure 7.7 top row) with the captured garment mask. Our 2D registration is based on our 3D non-rigid registration code for thin shell models [Li *et al.*, 2015a], and can deal with garment masks that usually have curved contours.

We first initialize a distance field using the segmented mask from the previous step. The category of the garment can be easily recognized by using a template matching algorithm, which leads to an associated garment initial template. We register the template with the garment mask by minimizing the following energy function:

$$E_T(S, \bar{S}) = E_{\text{fit}}(S, T) + E_{\text{def}}(S, \bar{S}), \tag{7.9}$$

where T is the target garment mask, S is the current deformed 2D contour, and $\bar{S}$ is a reference 2D contour, initialized to the template. $E_{\text{fit}}(S, T)$ penalizes discrepancies between the contour and the target mask, and $E_{\text{def}}(S, \bar{S})$ seeks to limit and regularize the deformation of the contour in order to preserve the angle features.

The term

$$E_{\text{fit}}(S, T) = \int_S (\text{dist}_T(\mathbf{x}))^2 \, dl, \tag{7.10}$$

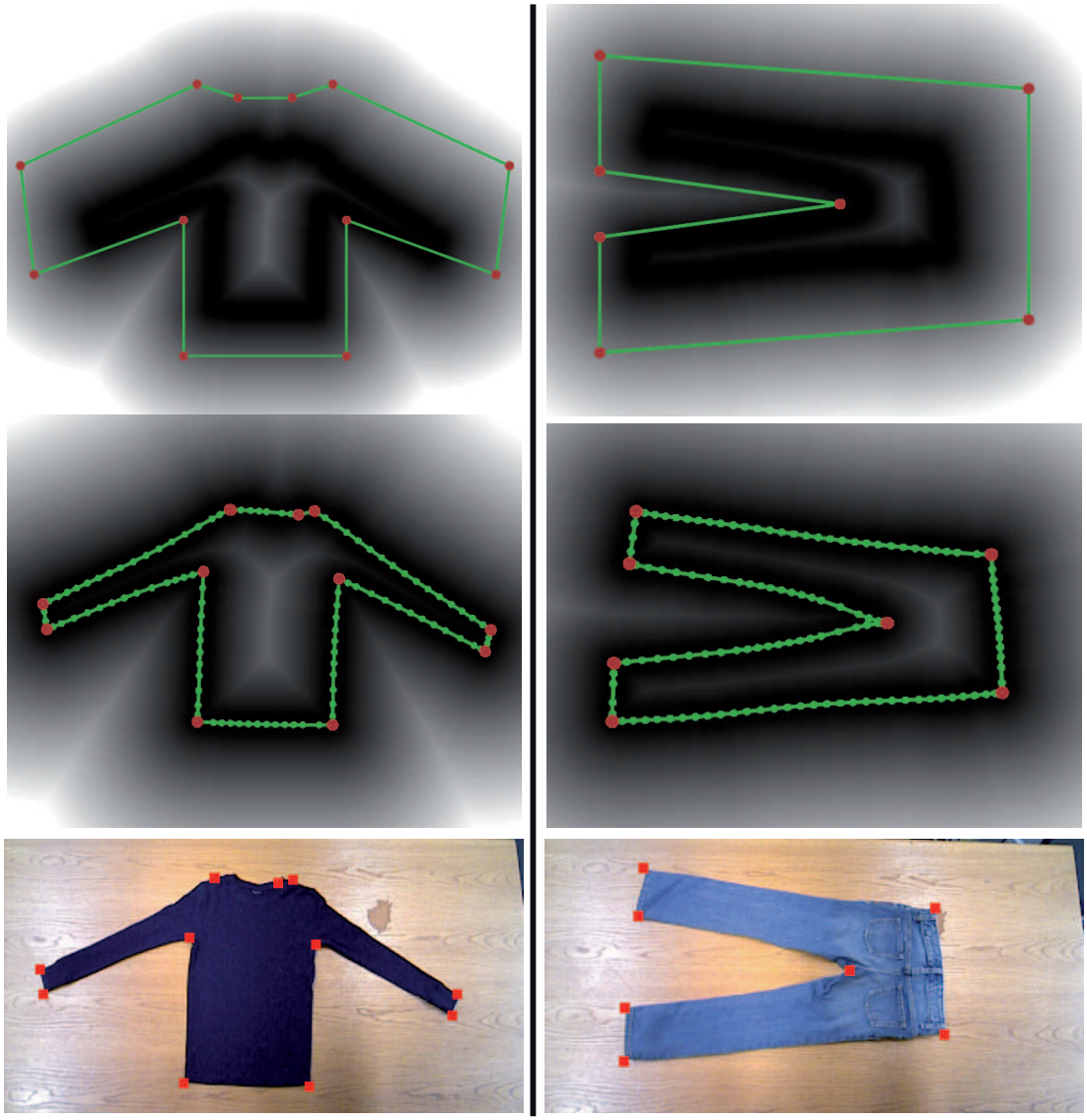


Figure 7.7: Red dots are the predefined key points for the garment, such as sleeve ends. Left column is a long-sleeve t-shirt example, and the right column is a pants example. TOP: Initialized contour. Each garment category is initialized with a different contour. MIDDLE: Fitting results. The contour shrinks onto the boundary of the garment according to the distance field. BOTTOM: Fitting results mapped back to the original input image.

integrates the distance value (dist_T) over the contour, and penalizes the deviation from the zero set

of the distance field. The term

$$E_{\text{def}}(S, \bar{S}) = \kappa E_{\text{stretch}}(\bar{S}, \mathbf{x}) + \beta E_{\text{bend}}(\bar{S}, \mathbf{x}), \tag{7.11}$$

accounts for the stretching and bending energies, weighted by user specified coefficients κ and β. E_{stretch} penalizes the change in the length given by

$$E_{\text{stretch}}(\bar{S}, \mathbf{x}) = \int_{\bar{S}} \frac{1}{2} \left(\frac{dl(\mathbf{x})}{d\bar{l}} - 1 \right)^2 d\bar{l}, \tag{7.12}$$

where $\bar{l}$ and l are the lengths of the reference and deformed contours.

E_{bend} penalizes the deviation from the prescribed curvature, given by

$$E_{\text{bend}}(\bar{S}, \mathbf{x}) = \int_{\bar{S}} \frac{1}{2} \left(\omega(\mathbf{x}) - \bar{\omega}(\mathbf{x}) \right)^2 d\bar{l}, \tag{7.13}$$

where $\bar{\omega}$ and ω are the curvatures of the reference and deformed contours.

We represent the contour as a closed loop consisting of line segments. In this discrete representation, E_{fit} is computed as

$$\tilde{E}_{\text{fit}}(S, T) = \sum_i \left(\text{dist}_T(\mathbf{x}_i) \right)^2 l_i, \tag{7.14}$$

where l_i and $\mathbf{x}_i$ are the length and the midpoint of the i-th line segment of the deformed contour. E_{stretch} is computed as

$$\tilde{E}_{\text{stretch}}(\bar{S}, \mathbf{x}) = \frac{1}{2} \sum_i \left(\frac{l_i}{\bar{l}_i} - 1 \right)^2 \bar{l}_i, \tag{7.15}$$

where $\bar{l}_i$ is length of the i-th line segment of the reference contour. E_{bend} is computed as

$$\tilde{E}_{\text{bend}}(\bar{S}, \mathbf{x}) = \frac{1}{2} \sum_j \left(\frac{\theta_j}{\bar{\theta}_j} - 1 \right)^2 \bar{L}_j, \tag{7.16}$$

where θ_j and $\bar{\theta}_j$ are the angles between the adjacent line segments at j-th vertex of the deformed and the reference contours, respectively. $\bar{L}_j$ are the average length of the adjacent line segments at j-th vertex of the reference contour.

Our registration iteratively updates S by using the secant version of the Levenberg-Marquardt algorithm [Madsen *et al.*, 2004][Nocedal and Wright, 2006]. Initially, each vertex in the template polygon is marked as a *feature point* (red points in Figure 7.7, top row), and is assigned a unique ID to identify its semantic meaning. At the end of each step, we subdivide each contour segment if

its length is larger than a threshold, and mark the newly added vertex as *non-feature points* (green points in Figure 7.7, middle row). Next, we merge any pair of adjacent segments if they do not share a feature point and their length is below a threshold. The subdivide and merge operations guarantee that the garment contour is sufficiently but not overly sampled. Then, we update the reference contour $\bar{S}$ by S.

We repeat the iteration until the reduction in $E_T(S, \bar{S})$ becomes sufficiently small. Finally, the positions as well as the semantic meanings of the feature points in the garment mask are identified by retrieving the feature points via the unique point ID in the registered contour.

7.4 Experimental Results

To evaluate our results, we tested our method on several different garments such as long-sleeve t-shirts, pants, and towels for multiple trials, as shown in Figure 7.10 left. These garments require both single and dual-arm folds. A high resolution video of our experimental results is online at http://www.cs.columbia.edu/~yli/laundry_robot.

7.4.1 Robot Setup

In our experiments, we use a Baxter research robot, which is equipped with two arms with seven degrees of freedom. We mount a Prime Sense Xtion range sensor [PrimeSense, 2013] on top of the Baxter head panel, which has been calibrated to the robot base frame, as shown in Figure 7.8, left. To improve grasp stability and form a closed loop controller, we add tactile sensors to the grippers, see Figure 7.8, right.

7.4.2 Measurement of parameters

To make the off-line simulation better approximate the real scenario, as described in Section 7.1, we manually measure the stretch resistance of each garment and friction on the table.

Figure 7.10, left shows a picture of all the test garments we used in different colors, sizes, and material properties. Figure 7.10, right table shows the measured parameters of each test garment, including stretch percentage and Friction angle, and corresponding *Maya* parameters. For common garments, these parameters do not have a significant variance. Therefore, we suggest that if re-

Figure 7.8: LEFT: Prime Sense mounted on the Baxter head panel. RIGHT: Gripper with tactile sensors.

searchers use simulators such as *Maya*, the average values of each column are a reasonably good initialization.

Figure 7.9: A picture of our test garments

7.4.3 Solution Space

The solution space is a subspace of the trajectory space where the folded garment ends in a shape with a dissimilarity score less than a threshold. Intuitively, a number of trajectories within the solution space will fold the garment, leaving its shape close to the desired shape. Figure 7.11

Garment Type	Stretch (%)	Friction Angle (°)	Maya Shear Resistance	Maya Friction
Long-Sleeve T-Shirt (large)	2.9	24.3	200	0.7
Long-Sleeve T-Shirt (small)	2.9	24.7	200	0.7
Jeans	2.9	19.1	200	0.5
Pants	1.7	21.9	340	0.6
Large Towel	2.2	18.7	260	0.5
Medium Towel	3.1	22.3	190	0.6
Small Towel	1.1	24.3	530	0.7
Average	**2.4**	**22.2**	**274**	**0.6**

Figure 7.10: Results for each unfolding test on the garments. We show the results of stretch percentage, Friction angle of the table, and the corresponding parameters in Maya by each test. The last row shows the average of each measurement component.

shows three sets of different trajectories for the second step of the towel folding. The color of the trajectory curve represents the dissimilarity between its folded shape and the desired folded shape (see color bar). We have found that trajectories within the solution space can vary to a degree while still allowing the robot to accomplish the folding task. This result also agrees with the fact that people do not have to follow a unique trajectory to fold the garment. However, trajectories outside the solution space cause issues for the folding task (see Figure 1.2). Our trajectory optimization automatically avoids such cases.

Another notable finding is that the dissimilarity is not symmetric in terms of the trajectory shape. For example, the middle plot has larger dissimilarities than the right plot in Figure 7.11. This is mainly caused by the friction. The robot should raise the starting point to a high enough position at the beginning to prevent the grasped portion of the towel pushing the other portion on the table. Due to this fact, an optimal trajectory usually does not have a symmetric shape. Intuitively, our optimizer will drive the height of the trajectories to a reasonable distance from the garment.

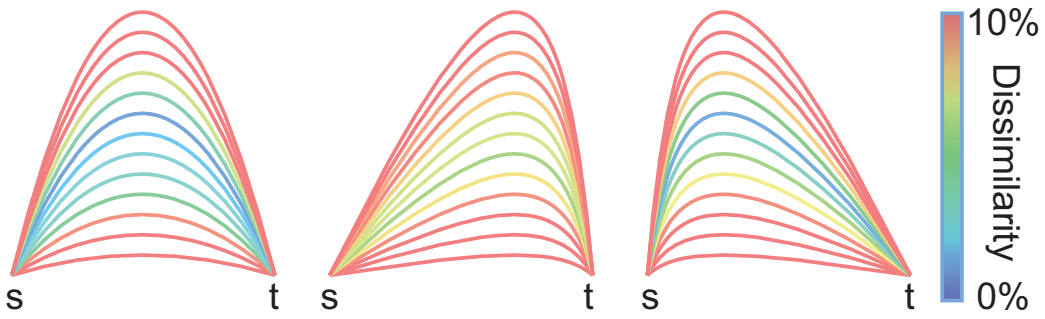


Figure 7.11: The dissimilarity values from different trajectories for folding the towel model in the second folding step. The trajectory is projected to a 2D plane for illustration purposes. S and T stand for the start and target position, respectively. (Best viewed in color)

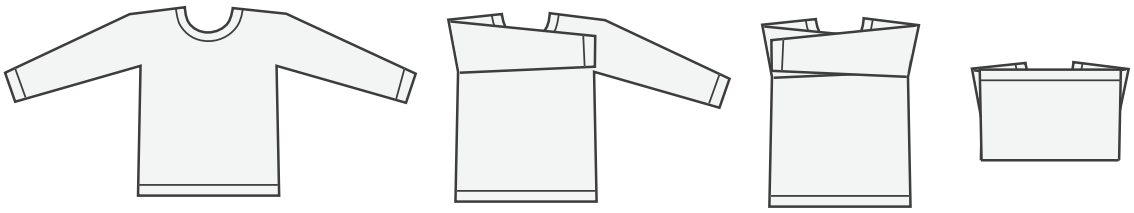


Figure 7.12: Garment folding plan for a long-sleeve T-shirt.

7.4.4 Garment manipulation and folding

Figure 7.13, Figure 7.14, and Figure 7.15 shows three successful folding examples from the simulation and the real world, including a long-sleeve shirt, a pair of pants, and a medium size towel. We show six key frames for each folding task. The folding poses from the simulation are in the first row of each group with an optimized trajectory. We also show corresponding results from the real world. The green tape contour on the table indicates the original position of the garment.

Each garment is first segmented from the background and key points are detected from the binary mask. Given the key points, a corresponding multi-step folding plan is created (The folding plan is predefined, and one of our folding plans for a long-sleeve T-shirt is shown in Figure 7.12). For each garment, we have optimized trajectories for each folding step. Here, we map these optimized trajectories to our scenario according to the generated folding plan. Then the Baxter robot follows the folding plan with optimized trajectories to fold the garment. We can see that the deformation of the real garment and the simulated garment is very similar. Therefore, the final folding

outcome is comparable to the simulation.

Table 7.1 shows statistical results of the garment folding test. Each time one or two robotic arms fold the garment counts as one fold. We run 10 trials for each test garment. It turns out that the folding performance of the Long-Sleeve T-Shirts and Towels are very stable with our optimized trajectories. Jeans and pants are less stable because the shear resistance of the surface is relatively high, and sometimes is difficult to bend, leading to unsuccessful folding. In the successful folding cases for jeans and pants, we sometimes ended up with small wrinkles, but the folding plan was still able to complete successfully. We also show the average time to fold a garment in the last row. The robot is able to fold most garments in about 1.5 minutes.

There is a trade-off between doing contour fitting at each step and total time spent to fold a garment. In this work, we start with one template and then assume that each step after that the folded garment is close to that in the simulation. Our experimental results as shown in Table 7.1 verify that this method works well and is able to save time since we only do the contour fitting once. With our simulated trajectories, the Baxter robot is able to fold a garment under predefined steps correctly. An alternative method could use the contour fitting at each step but this would require more time and computation.

We note that some failures due to the motor control error from the Baxter robot. When the robot executes an optimized trajectory, its arm suffers from a sudden drop or jitter. Such actions will raise pull forces to the garment, leading to drift and inaccurate folding. This can be solved by using an industrial level robotic arm with more accurate control. We also note that failures can be recognized with the correct sensing suite, and we are currently investigating ways to effect online error recovery for such failures. One difference between the simulation and the real world we found is that moving a point on the mesh in the simulation is different from using a gripper to grasp a small area of a real garment and move it. In the future, we hope to be able to simulate a similar grasp effect for the trajectory optimization.

7.5 Summary

In this chapter, we propose a novel solution to find an optimal trajectory for manipulation of deformable garments. We first create a simulation environment that is comparable to the real world.

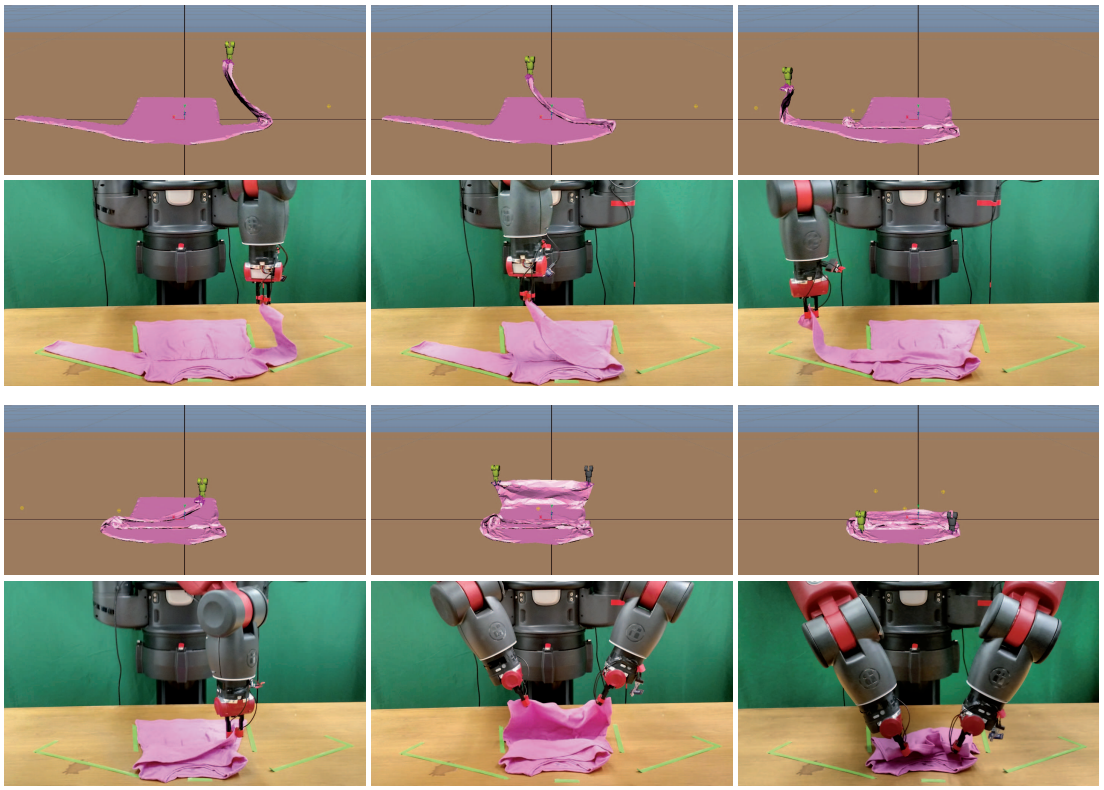


Figure 7.13: Successful folding examples with optimized folding trajectories from off-line simulation. The first row of each group is from the simulation and the second row is from the real world (Green tape shows the original garment contour position). This is a long-sleeve shirt folding with 3 steps.

By minimizing a quadratic objective function that measures dissimilarity between simulated folded shape and user specified shape iteratively, we obtain an optimized trajectory. The trajectory is then mapped to a real robot and executed accordingly. Experimental results demonstrate that with our optimized trajectories, the Baxter robot can manipulate the garment efficiently and accurately.

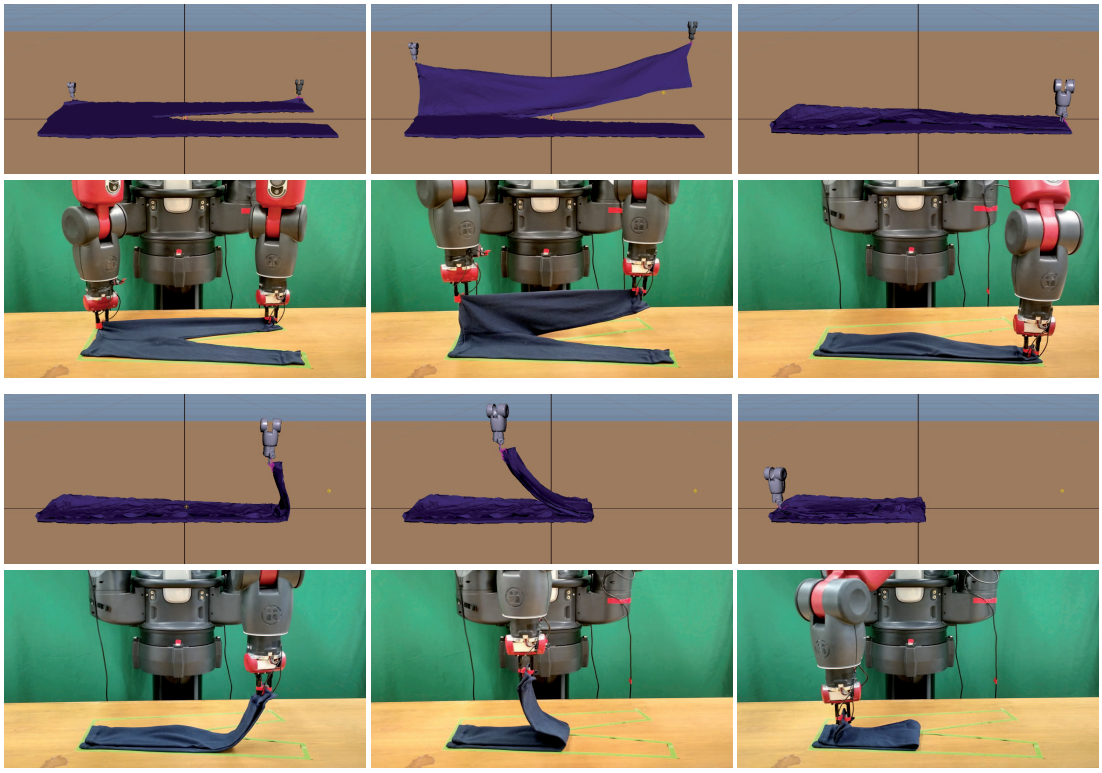


Figure 7.14: Successful folding examples with optimized folding trajectories from off-line simulation. The first row of each group is from the simulation and the second row is from the real world (Green tape shows the original garment contour position). This is a Long pants folding with 2 steps.

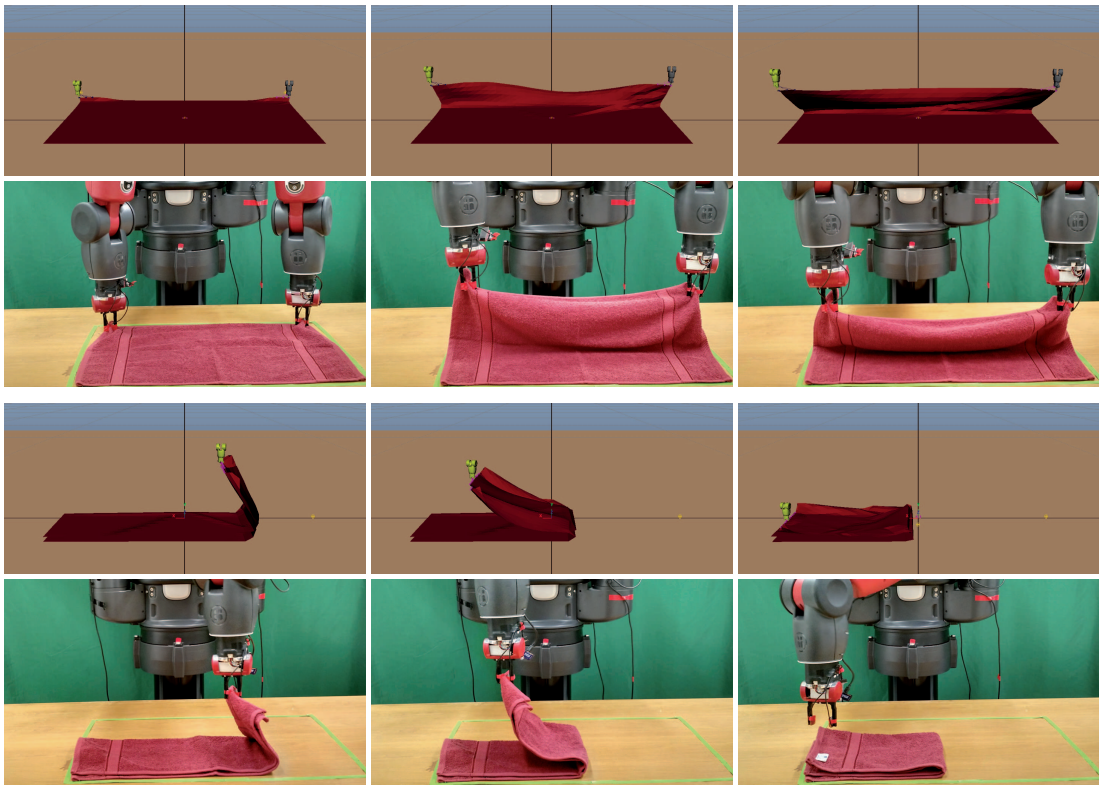


Figure 7.15: Successful folding examples with optimized folding trajectories from off-line simulation. The first row of each group is from the simulation and the second row is from the real world (Green tape shows the original garment contour position). This is a medium size towel folding with 2 steps.

Garment Type	# of folds	Success Rate	Avg. Time (sec)
L-S T-Shirt (large)	3	10/10	121
L-S T-Shirt (small)	3	10/10	118
Jeans	2	7/10	88
Pants	2	8/10	88
Large Towel	2	10/10	90
Medium Towel	2	10/10	88
Small Towel	2	10/10	83
Average	**2.3**	**9.3/10**	**97**

Table 7.1: Results of folding test for each garment . We show the number of folding steps, successful rate, and total time of each garment. Each garment has been tested 10 times. L-S stands for Long-Sleeve. The time is the average over all successful trials for each garment.

Chapter 8

Conclusion

8.1 Summary of Contributions

Robotic manipulation of deformable objects is a challenging problem especially because of the complexity of the many different ways an object can deform. Searching within such a high dimensional state space makes it difficult to recognize, track, and manipulate deformable objects. In this thesis, we presented a *feed-forward, model-driven* approach to address this challenge, using a pre-computed, simulated database of deformable object models. The models are detailed, robust, and easy to construct, and the physics engine can predict accurately in simulation the behavior of the objects, which can then be used in real physical setting. This will bridge the gap between the simulation world and the real world. The *feed-forward, model-driven* approach takes advantages of the simulation and generates a large number of instances for learning approaches, which not only alleviates the burden of the data collection, but also impels domain adaptation of the presented methods to other applications easier and faster.

Releasing a simulation database to support the presented methods can be beneficial to the research community. Mesh models of common deformable garments are simulated with the garments picked up in multiple different poses under gravity, and stored in an indexable database for fast and efficient retrieval. To validate this approach, we developed a comprehensive pipeline for manipulating clothing as in a typical laundry task. First, the database is used to estimate poses of garments in an arbitrary positions. A full featured 3D volumetric model of the garment is constructed in real-time and volumetric features are then used to obtain the most similar model in the database to

predict the object category and pose. Second, the database can significantly benefit the manipulation of deformable objects via non-rigid registration, providing accurate correspondences between the reconstructed object model and the database models. Third, the accurate model simulation can also be used to optimize the trajectories for manipulation of deformable objects, such as the folding of garments. Extensive experimental results are shown for the tasks above using a variety of different clothing. To further support the research in this or related fields, we release the script for the entire simulation in Maya that we used to create the database. The simulation can be easily adapted to the new garment mesh models by giving the semantic grasping points.

With the recent fast development of low-cost depth sensors, such as the Kinect sensor, a huge demand of processing multi-channel sensor data is being created. In our pipeline, using the depth sensor, we achieved significant improvements of the recognition, in terms of accuracy and speed, and spread out in the following manipulation tasks. The real-time 3D reconstruction and recognition are an excellent example. Another application of the depth sensor is multi-sensor fusion of vision sources for classification of wrinkles, which tackles a few difficult scenarios in robotic ironing.

Finally, within our comprehensive pipeline for manipulating clothing, the interactions between computer vision, machine learning, computer graphics, and robotics will develop a set of new research topics. Meanwhile, the inter-disciplinary approaches will make the entire research process equally comfortable in the physical and computational world.

8.1.1 Recognition

Using Depth Perception

We first designed an automatic simulation system that can capture 90 depth images from different view points of a target object in 20-50 different poses. Then we propose a method for estimating deformable object pose using these simulation results and dictionary learning via spatial pyramid matching and sparse coding. We further formulation of the pose recognition problem as a hierarchical image classification task that uses depth images regardless of complicated texture on the object. Compared to the RGB color space, the depth space has much lower complexity so that it is not only require less data for the training, but also more robust to the texture on the objects. Experimental results in simulation, with real clothing, and also with a physical robot on a variety of deformable objects including sweaters, jeans, and short pants in different sizes. The results show

that our approach is robust to estimate the object pose and to facilitate regrasping and folding tasks.

Using Volumetric Approach

For the volumetric approach, we present a real-time approach to reconstruct a smooth 3D model of a moving (e.g. rotating) deformable object from noisy background without user interaction. This method is much faster than the previous one with the depth perception. Specifically, we applied 3D segmentation on the depth images of a rotating deformable object, which are captured from a Kinect sensor. Then these segmented depth images are passed into the Kinect algorithm, which has an assumption that the object is not moving. With the reconstructed 3D model, we formulate the pose recognition problem as a real-time 3D shape matching task with a learned distance metric. Experimental results with many different garments that show improved accuracy over our previous method using depth image classification, and orders of magnitude speed up yielding real-time performance for the pose estimation task.

8.1.2 Regrasping and Unfolding

In this part, we propose a novel solution for the problem of unfolding a garment to a desired configuration via regrasping. Specifically, we introduced a constrained weighted metric for evaluating grasping points during regrasping, which can also be used for a convergence criterion. By using a fast, two-stage deformable object registration algorithm (rigid and non-rigid registration)that integrates off-line simulated results with online localization, we can find correspondence between the database mesh and the reconstructed mesh. We also present a novel method for analysis of local curvature using IR scan data for stable grasp. The IR data is used for computing the local curvature and looking for a surface bump that can fit into the gripper opening. Experimental results showing that our method applied on a Baxter robot is able to recognize a garment from an arbitrary pose, successfully regrasp it to put it into a known configuration, and place the garment flat on a table, which can then be folded in a subsequent process.

8.1.3 Ironing

When the garment is laid flat on a table, there are always some wrinkles on it. However, some of them can be removed by pulling the boundary of the garment. Therefore, we introduce a taxonomy of common wrinkles on a piece of cloth and garments, as well as the physical-level explanation. To

find the right wrinkles, we present a multi-sensor probabilistic framework for classifying detected wrinkles for robotic ironing, which combines curvature-based and discontinuity-based images. Experiments with a Baxter robot showing effective robotic ironing using both static and dynamic ironing.

8.1.4 Folding

To fold the garment efficiently, the trajectories of the folding are very important. Bad trajectories will cause the garment to move or create more wrinkles. We would like to use the off-line simulation to find the best trajectories. We first applied a novel approach that adjusts the simulation environment to the robot working environment for the purpose of creating a similar manipulation result. Then under this simulation environment, by using an online optimization algorithm that learns optimal trajectories for manipulation from mathematical model evolution combined with predictive thin shell simulation, we found the optimized trajectories for a specific folding action. These trajectories are general in that they can be scaled to accommodate similar garments of different size. To mapping the trajectories in the simulation to the real world garments, we present a fast and robust algorithm that can detect garment key points such as sleeve ends, collar, and waist corner, automatically. These key points can be used for folding plan generation. Experimental results with a Baxter dual-arm robot showing successful folding trajectories for a number of different garments including sweaters, pants, and towels

8.2 Future Work

8.2.1 Recognition

Our work can be extended in several directions. First, it is possible to add the color and texture of garments as features to further improve the recognition accuracy. Second, an accurate and fast method for pose estimation of deformable objects may benefit a variety of practical applications such as clothes folding, which will be much easier once the robot has an accurate mesh model and knows which point it is grasping. Last but not least, while this work is about vision techniques benefiting robotics applications, it is also possible to use robotics to simplify vision tasks. For example, simple actions such as shaking from robotic arms may help resolve the visual variance from friction.

Though we use a relatively small test set for our experiments, we notice that our models can also be generalized to recognize similar but unseen garments. For example, long-sleeve shirts and jackets can be considered as similar garments to our sweater model. Also, knit pants and suit pants are similar to our jeans model. Although they are made of different materials, the way they deform are similar to our training models in some poses. One direction is to further explore the performance of more garments using our method such as a jacket, a pair of long sports pants, etc. For all the recognition tasks, we used a Maya simulated database. The database contains 37 different garments, including Long-sleeve T-Shirt, T-shirt, pants, shorts, and socks, etc. We will further clean up, re-organize, and publish the database online to facilitate other researchers in this field, with detailed documents and necessary scripts.

8.2.2 Unfolding

In spite of the promising results shown, there is still some space for improvement. One observation is that the regrasping of the deformable objects is not very stable in terms of the local curvature. Our current solution is to scan the surface of the object and approach a relatively desired location to grasp, which might fail if all the locations are unsatisfactory. One possible solution is to rotate the garment, and re-scan the local curvature. Also, using a pair of sharper head grippers may also help. Another observation is that even when we grasp at the desired position, there exists some twists, such as the arm part of a sweater. This will bring additional wrinkles when the garment is being laid

flat. One possible solution is to shake the garment several times to let it to be maximumly stretched before regrasping.

8.2.3 Ironing

Robotic ironing is a very challenging task. A full solution to the problem requires complex surface analysis, regrasping, and hybrid force/position control of the iron. In this part, we have addressed the surface analysis problem and also used a position controlled robotic arm to implement ironing. While position control can remove many wrinkles, we also believe that with more accurate force control, the effects of ironing can be improved. One of our future directions is to add the force control component to our system. The Lambertian assumption used in the discontinuity scans works for many garments but it is not clear it will work for all garments.

8.2.4 Folding

There are many challenges during the robotic folding. One of them is that the grasped part may drop in the middle. Then how to recover from this state becomes an important issue. We can start from the current state which may require addition image analysis. Or we can start from the beginning (e.g. lay flat) which requires additional time and manipulations. Another challenge is how to extend the current folding plan to novel garments without additional simulation for optimized folding trajectories. One possible solution is using a regression model to learn the correspondence between the material properties and the trajectories.

Part III

Bibliography

Bibliography

[Argall *et al.*, 2009] B. D Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[Belongie *et al.*, 2002] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, 24(24):509–522, Apr 2002.

[Besl and McKay, 1992] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, Feb 1992.

[Brogrdh, 2007] T. Brogrdh. Present and future robot control developmentan industrial perspective. *Annual Reviews in Control*, 31(1):69 – 79, 2007.

[Calinon *et al.*, 2010] S. Calinon, I. Sardellitti, and D. G. Caldwell. Learning-based control strategy for safe human-robot interaction exploiting task and robot redundancies. In *Proc. IROS*, 2010.

[Chen *et al.*, 2013] J. Chen, D. Bautembach, and S. Izadi. Scalable real-time volumetric surface reconstruction. *SIGGRAPH*, 32(4):113:1–113:16, jul 2013.

[Cusumano-Towner *et al.*, 2011] M. Cusumano-Towner, A. Singh, S. Miller, J. F. OBrien, and P. Abbeel. Bringing clothing into desired configurations with limited perception. In *Proc. ICRA*, 2011.

[Dai *et al.*, 2004a] J. S. Dai, P. M. Taylor, H. Liu, , and H. Lin. Folding algorithms and mechanisms synthesis for robotic ironing. In *International Journal of Clothing Science and Technology*, 2004.

[Dai *et al.*, 2004b] J. S. Dai, P. M. Taylor, P. Sanguanpiyapan, and H. Lin. Trajectory and orientation analysis of the ironing process for robotic automation. In *International Journal of Clothing Science and Technology*, 2004.

[Doumanoglou *et al.*, 2014] A. Doumanoglou, A. Kargakos, T. K. Kim, and S. Malassiotis. Autonomous active recognition and unfolding of clothes using random decision forests and probabilistic planning. In *Proc. ICRA*, May 2014.

[Fan *et al.*, 2008] R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin. Liblinear: A library for large linear classification. *JMLR*, 9:1871–1874, 2008.

[Farin, 1988] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, 1988.

[Fei-Fei and Perona, 2005] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. *CVPR*, 2005.

[Felzenszwalb *et al.*, 2010] P. F Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. PAMI*, 32:1627–1645, July 2010.

[Frome *et al.*, 2004] A. Frome, D. Huber, R. Kolluri, T. Blow, and J. Malik. Recognizing objects in range data using regional point descriptors. In *Proc. ECCV*, pages 224–237, 2004.

[Goldfeder *et al.*, 2009] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen. The Columbia grasp database. *Proc. ICRA*, 2009.

[Grauman and Darrell, 2005] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. *ICCV*, 2005.

[Grinspun *et al.*, 2003] E. Grinspun, A. N. Hirani, M. Desbrun, and P. Schröder. Discrete shells. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, pages 62–67, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[Huttenlocher *et al.*, 1993] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing images using the Hausdorff distance. *PAMI*, 1993.

[Joachims, 2002] T. Joachims. Optimizing search engines using clickthrough data. In *Proc. KDD*, pages 133–142, 2002.

[Kita and Kita, 2002] Y. Kita and N. Kita. A model-driven method of estimating the state of clothes for manipulating it. In *Proc. WACV*, 2002.

[Kita *et al.*, 2011a] Y. Kita, F. Kanehiro, T. Ueshiba, and N. Kita. Clothes handling based on recognition by strategic observation. In *Humanoid Robots*, 2011.

[Kita *et al.*, 2011b] Y. Kita, T. Ueshiba, E-S Neo, and N. Kita. Clothes state recognition using 3D observed data. In *Proc. ICRA*, 2011.

[Koenderink and van Doorn, 1992] J. J Koenderink and A. J van Doorn. Surface shape and curvature scales. *Image and Vision Computing*, 10(8):557 – 564, 1992.

[Kormushev *et al.*, 2011] P. Kormushev, S. Calinon, and D. G. Caldwell. Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input. In *Advanced Robotics*, 2011.

[Latecki *et al.*, 2000] L. Latecki, R. Lakamper, and T. Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. In *Proc. CVPR*, 2000.

[Lazebnik *et al.*, 2006] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *CVPR*, 2006.

[Le *et al.*, 2013] T-H-L Le, M. Jilich, A. Landini, M. Zoppi, D. Zlatanov, and R. Molfino. On the development of a specialized flexible gripper for garment handling. *Journal of automation and control engineering*, 1(3), 2013.

[Lee *et al.*, 2007] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. *Proc. NIPS*, pages 801–808, 2007.

[Li *et al.*, 2008] H. Li, R. W. Sumner, and M. Pauly. Global correspondence optimization for non-rigid registration of depth scans. In *Proceedings of the Symposium on Geometry Processing*, SGP '08, pages 1421–1430, Aire-la-Ville, Switzerland, 2008.

[Li *et al.*, 2013] H. Li, E. Vouga, A. Gudym, L. Luo, J. T. Barron, and G. Gusev. 3D self-portraits. *SIGGRAPH Asia*, 32(6), Nov. 2013.

[Li *et al.*, 2014a] Y. Li, C-F Chen, and P. K. Allen. Recognition of deformable object category and pose. In *Proc. ICRA*, June 2014.

[Li *et al.*, 2014b] Y. Li, Y. Wang, M. Case, S-F Chang, and P. K. Allen. Real-time pose estimation of deformable objects using a volumetric approach. In *Proc. IROS*, September 2014.

[Li *et al.*, 2015a] Y. Li, D. Xu, Y. Yue, Y. Wang, S-F Chang, E. Grinspun, and P. K. Allen. Regrasping and unfolding of deformable garments using predictive thin shell modeling. In *Proc. ICRA*, May 2015.

[Li *et al.*, 2015b] Y. Li, Y. Yue, D. Xu, E. Grinspun, and P. K. Allen. Folding deformable objects using predictive simulation and trajectory optimization. In *Proc. IROS*, 2015.

[Liu *et al.*, 2008] L. Liu, L. Zhang, Y. Xu, C. Gotsman, and S. J. Gortler. A local/global approach to mesh parameterization. In *SGP*, 2008.

[Lowe, 1999] D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. ICCV*, 1999.

[Lowe, 2004] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91 – 110, 2004.

[Madsen *et al.*, 2004] K. Madsen, H. B. Nielsen, and O. Tingleff. Methods for non-linear least squares problems (2nd ed.). Technical report, Technical University of Denmark, 2004.

[Mairal *et al.*, 2009] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. *Proc. ICML*, 2009.

[Mairal *et al.*, 2010] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *JMLR*, 11:19 – 60, 2010.

[Maitin-Shepard *et al.*, 2010] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel. Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In *Proc. ICRA*, 2010.

[Maya, 2014] 2014. Maya, http://www.autodesk.com/products/autodesk-maya/.

[Miller *et al.*, 2011] S. Miller, M. Fritz, T. Darrell, and P. Abbeel. Parametrized shape models for clothing. In *Proc. ICRA*, Sept. 2011.

[Miller *et al.*, 2012] S. Miller, J. Berg, M. Fritz, T. Darrell, K. Goldberg, and P. Abbeel. A geometric approach to robotic laundry folding. *IJRR*, 31(2):249–267, 2012.

[Newcombe *et al.*, 2011] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, pages 127–136, 2011.

[Nocedal and Wright, 2006] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 2006.

[Oren and Nayar, 1995] M. Oren and S. Nayar. Generalization of the Lambertian model and implications for machine vision. *International Journal of Computer Vision*, 14(3):227–251, 1995.

[Osada and Funkhouser, 2001] R. Osada and T. Funkhouser. Matching 3d models with shape distributions. In *Proc. SMI Int. Conf.*, 2001.

[Osawa *et al.*, 2007] Fumiaki Osawa, Hiroaki Seki, and Yoshitsugu Kamiya. Unfolding of massive laundry and classification types by dual manipulator. *Journal of Advanced and Intelligent Informatics*, 11(5), 2007.

[Pedersoli *et al.*, 2011] M. Pedersoli, A. Vedaldi, and J. Gonzalez. A coarse-to-fine approach for fast deformable object detection. *CVPR*, 2011.

[PrimeSense, 2013] 2013. Prime Sense, http://en.wikipedia.org/wiki/PrimeSense/.

[Ramisa *et al.*, 2012] A. Ramisa, G. Alenya, F. Moreno-Noguer, and C. Torras. Using depth and appearance features for informed robot grasping of highly wrinkled clothes. In *Proc. ICRA*, Sept. 2012.

[Ravishankar *et al.*, 2008] S. Ravishankar, A. Jain, and A. Mittal. Multi-stage contour based detection of deformable objects. *ECCV*, 2008.

[Rother *et al.*, 2004] C. Rother, V. Kolmogorov, and A. Blake. "Grabcut": Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, August 2004.

[R.Tsai, 2002] Y-H R.Tsai. Rapid and accurate computation of the distance function using grids. *Journal of Computational Physics*, 178(1):175 – 195, 2002.

[Schulman *et al.*, 2013] J. Schulman, A. Lee, J. Ho, and P. Abbeel. Tracking deformable objects with point clouds. In *Proc. ICRA*, 2013.

[Stria *et al.*, 2014a] J. Stria, D. Prusa, and V. Hlavac. Polygonal models for clothing. In *Proc. Towards Autonomous Robotic Systems*, 2014.

[Stria *et al.*, 2014b] J. Stria, D. Prusa, V. Hlavac, L. Wagner, V. Petrik, P. Krsek, and V. Smutny. Garment perception and its folding using a dual-arm robot. In *Proc. IROS*, Sept. 2014.

[Sun *et al.*, 2015] L. Sun, G. Aragon-Camarasa, S. Rogers, and P. Siebert. Accurate garment surface analysis using an active stereo robot head with application to dual-arm flattening. In *Proc. ICRA*, 2015.

[Tam *et al.*, 2013] G.K.L. Tam, Zhi-Quan Cheng, Yu-Kun Lai, F. C. Langbein, Yonghuai Liu, D. Marshall, R.R. Martin, Xian-Fang Sun, and P.L. Rosin. Registration of 3D point clouds and meshes: A survey from rigid to nonrigid. *Visualization and Computer Graphics, IEEE Transactions on*, 19(7):1199–1217, July 2013.

[Thayananthan *et al.*, 2003] A. Thayananthan, B. Stenger, P. H. S. Torr, and R. Cipolla. Shape context and chamfer matching in cluttered scenes. In *CVPR*, 2003.

[Toshev *et al.*, 2010] A. Toshev, B. Taskar, and K. Daniilidis. Object detection via boundary structure segmentation. In *Proc. CVPR*, 2010.

[Tu and Yuille, 2004] Z. Tu and A. Yuille. Shape matching and recognition: Using generative models and informative features. In *Proc. ECCV*, 2004.

[Umetani *et al.*, 2011] N. Umetani, D. M. Kaufman, T. Igarashi, and E. Grinspun. Sensitive Couture for Interactive Garment Editing and Modeling. *ACM Transactions on Graphics*, 30(4), Aug 2011.

[van den Berg *et al.*, 2010] J. van den Berg, S. Miller, K. Goldberg, and P. Abbeel. Gravity-based robotic cloth folding. In *Proc. Intl. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2010.

[Vedaldi and Fulkerson, 2008] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms, 2008. http://www.vlfeat.org/.

[Vincent and Soille, 1991] L. Vincent and P. Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991.

[Wang *et al.*, 2006] J. Wang, L. Yin, X. Wei, and Y. Sun. 3d facial expression recognition based on primitive surface feature distribution. In *Proc. CVPR*, 2006.

[Wang *et al.*, 2011] P-C Wang, S. Miller, M. Fritz, T. Darrell, and P. Abbbeel. Perception for the manipulation of socks. *Proc. IROS*, 2011.

[Wang *et al.*, 2013] Y. Wang, R. Ji, and S.-F. Chang. Label propagation from imagenet to 3d point clouds. In *Proc. CVPR*, June 2013.

[Willimon *et al.*, 2011] B. Willimon, S. Birchfield, and I. Walker. Classification of clothing using interactive perception. In *Proc. ICRA*, 2011.

[Willimon *et al.*, 2013] B. Willimon, I. Walker, and S. Birchfield. A new approach to clothing classification using mid-level layers. In *Proc. ICRA*, 2013.

[Wu *et al.*, 2008] C. Wu, B. Clipp, X. Li, J-M Frahm, and M. Pollefeys. 3d model matching with viewpoint-invariant patches. In *Proc. CVPR*, 2008.

[Yang *et al.*, 2009] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching uisng sparse coding for image classification. *CVPR*, 2009.

[Zhang *et al.*, 1999] R. Zhang, P-S Tsai, E. Cryer, and M. Shah. Shape-from-shading: a survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(8):690–706, 1999.

[Zhang, 2012] Zhengyou Zhang. Microsoft kinect sensor and its effect. *IEEE MultiMedia*, 19(2):4–10, April 2012.

Part IV

Appendix

Baxter robot

In our experiments, we used a Baxter robot built by Rethink Robot as shown in Figure 8.1. Baxter is a 190 cm tall (with pedestal), two-armed robot with an animated face. It weighs 306 lbs with the pedestal. It is designed for simple industrial jobs such as loading, unloading, sorting, and handling of materials. Each arm has 7 degrees of freedom with a changeable finger gripper, see Figure 8.1, right.

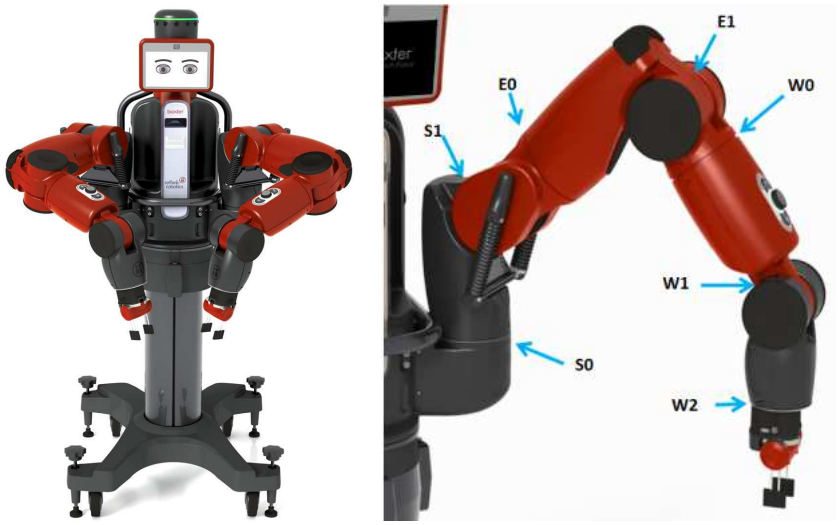


Figure 8.1: LEFT: The Baxter Robot. RIGHT: One arm with 7 joints annotated. The images are from Rethink Robotics website.

Tactile sensor

In order to achieve stable grasps, tactile sensors are attached to each of the Baxter grippers. Here, we used a tactile sensor from Takktile LLC called "TakkTile Kit for Hubo Robot", as shown in Figure 8.2. It has a relatively small sensor shape that can fit into our robot grippers, see Figure 5.6 and Figure 7.8. The reading of the sensor is in proportion to the pressure on it, which is in the format of a float number, ranged from 0 to 1000. When there is nothing inside a pair of the grippers in close mode, the reading is zero or very close to zero. However, when something stays inside in the close mode, the reading increases. Empirically, we define a stable grasp when the reading is above 0.6. If the reading is less than this number during the grasp, we will release the grippers and regrasp.

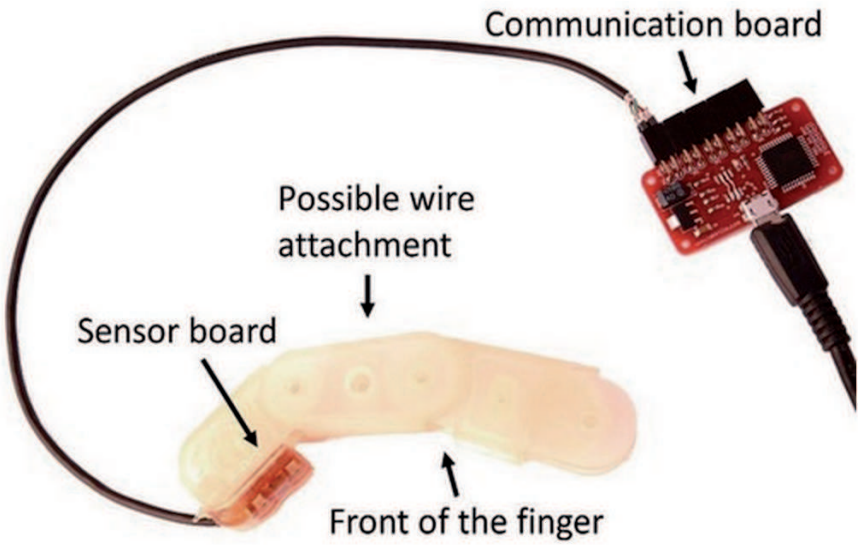


Figure 8.2: A picture of a sample tactile sensor attached on an object (a finger model), with its controller board (in red). The image is from Takktile LLC website.